IRIS R18 - User Guide

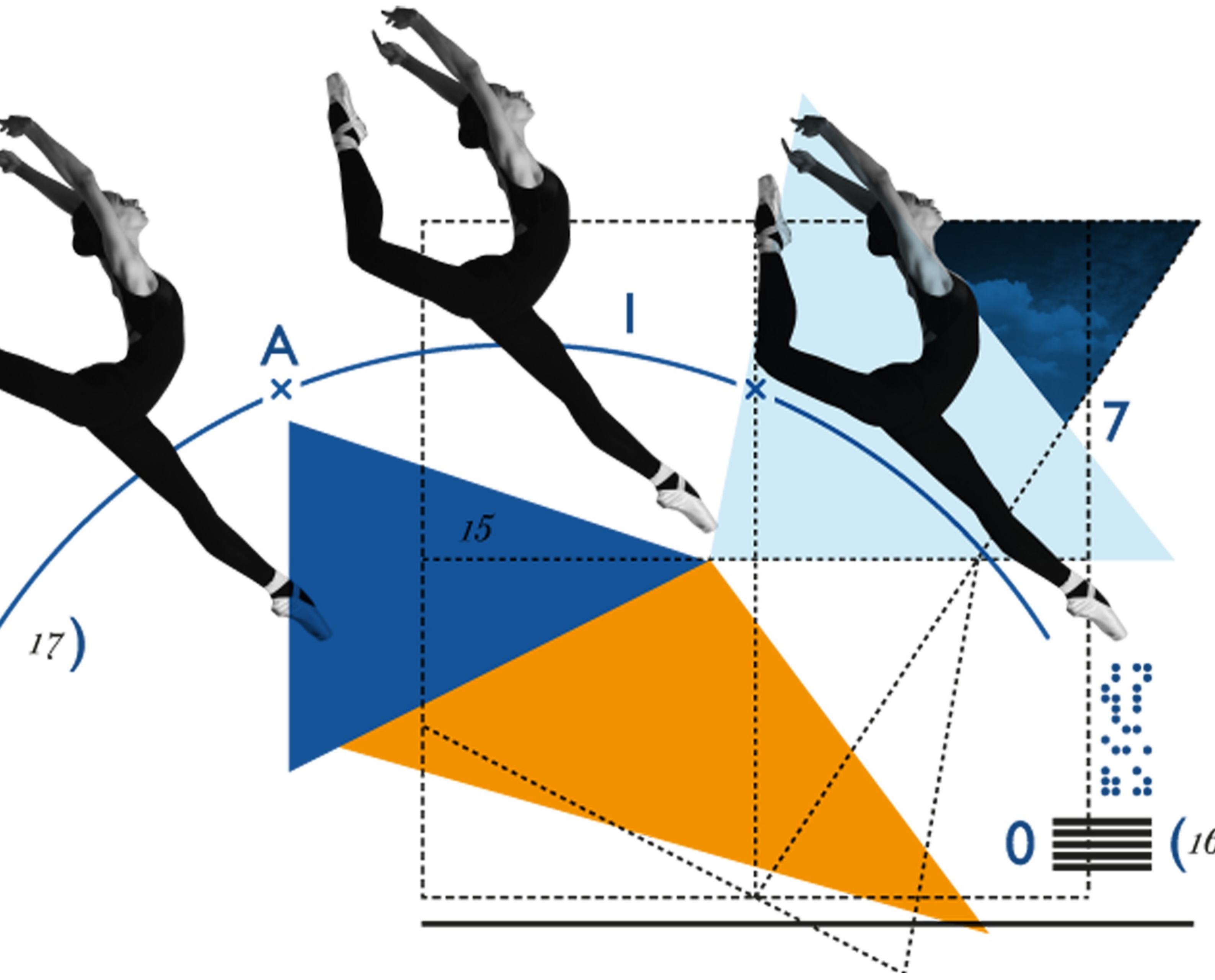


Table Of Contents

Introduction

Purpose of the Guide

This document helps the user to design models and resources for the data service which is generated from the designed models to communicate with T24. This allows the model to be used for all T24 users. This document also helps to configure and deploy IRIS R18 provider and publisher components of T24 infrastructure.

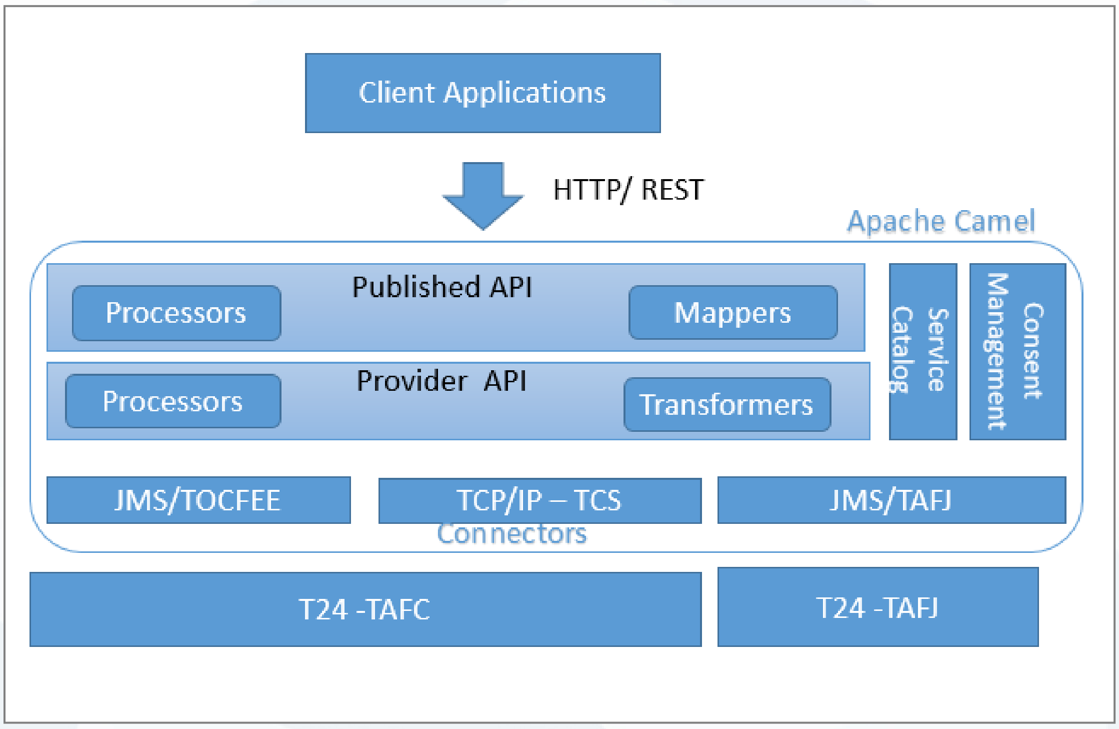
Intended Audience

This User Guide is intended for the use of External and Partner teams to deploy IRIS R18 Module.

Overview

IRIS R18 is a lightweight, REST standards based solution that uses OFS message format to communicate with T24. This allows the solution to be used for all T24 customers across release versions.

The below screenshot shows the overview of the solution and its components.



- Provider API Layer : hosts REST APIs for T24 modules
- Published API: hosts Published API. Used to transform provider API to an interface external applications or standards message formats.
- Connectors: Standard connectors available to connect to T24 based on the T24 deployment (TAFC/TAFJ).
- Consent Management: management layer to apply access control while servicing API requests from client application.
- Service Catalogue: hosts SWAGGER API documentation for Provider and Publisher APIs.
- Apache Camel: is an open source frameworks used to define rest end points, transforms data and route messages in Publisher and Provider layer.

Prerequisites

The following tools should be available for development and testing purpose:

Item	Link
Apache Camel	http://camel.apache.org/
Maven	https://maven.apache.org/
Spring	https://projects.spring.io/spring-framework/
Open API Spec, fka Swagger	https://swagger.io/
Postman	https://www.getpostman.com/
Eclipse	http://www.eclipse.org/
Junit	http://junit.org

Maven, Java

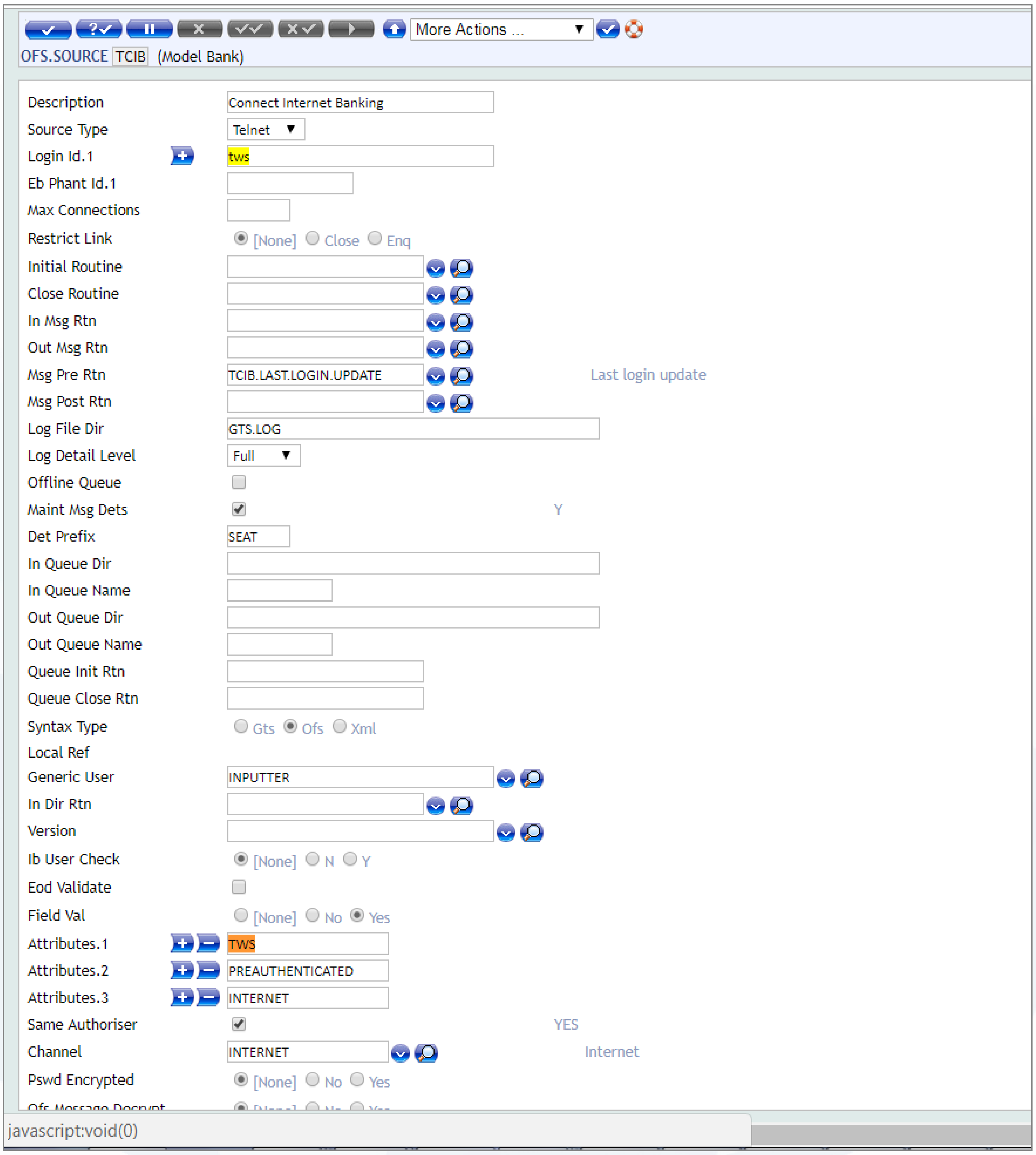
Maven and Java 8 are required. You can download the other dependencies from the Maven repository.

Key Application configuration

T24 Application Configuration

OFS.SOURCE Configuration

1. **FIELD Val** in order to enable Field validation for OFS Messages specifically for AA activities, configure the attribute **Field Val** as Yes of OFS.SOURCE

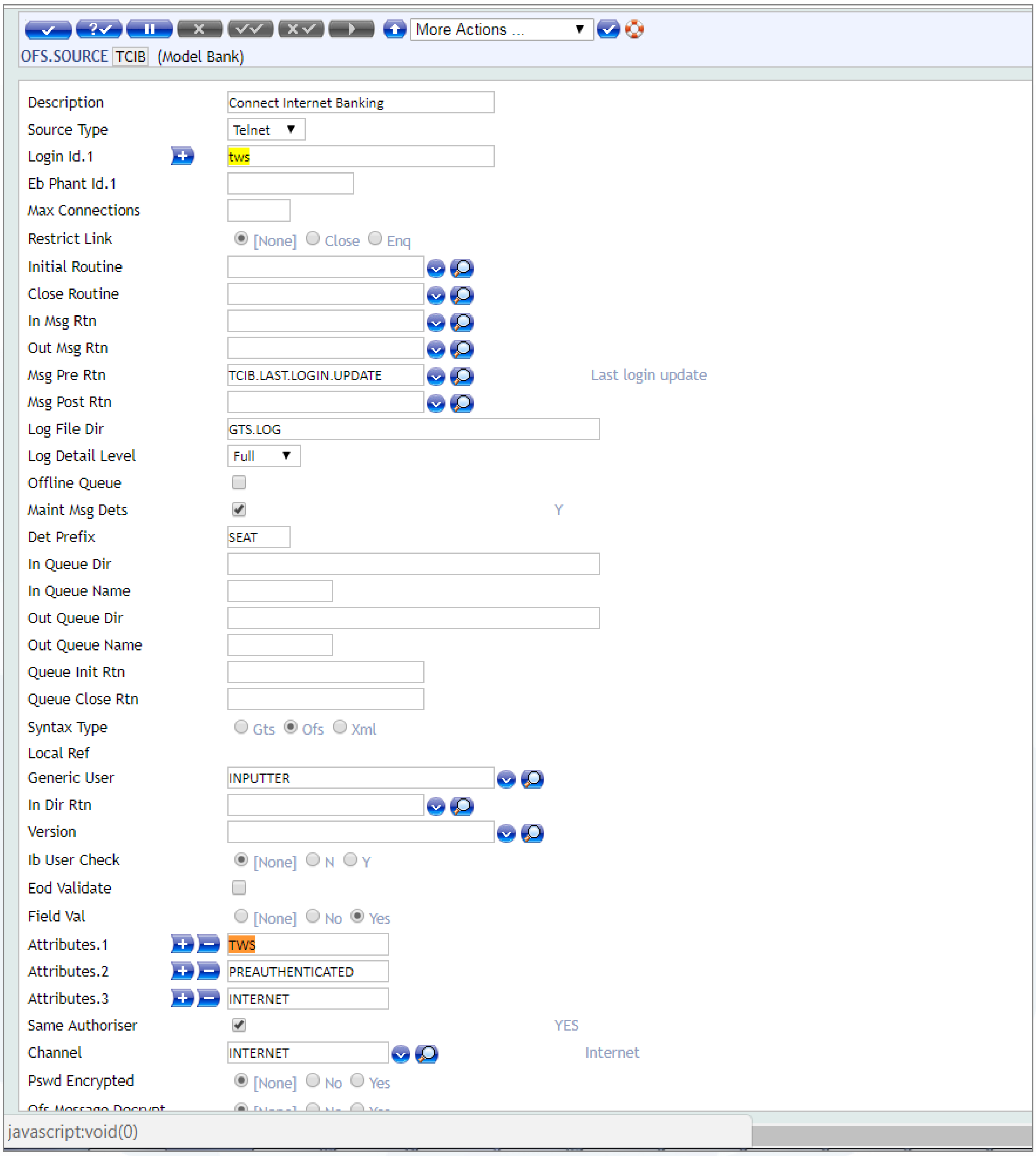


OFS.SOURCE TCIB (Model Bank)

Description	Connect Internet Banking	
Source Type	Telnet	
Login Id.1	tws	
Eb Phant Id.1		
Max Connections		
Restrict Link	[✓ [None]] Close Enq	
Initial Routine		
Close Routine		
In Msg Rtn		
Out Msg Rtn		
Msg Pre Rtn	TCIB.LAST.LOGIN.UPDATE	Last login update
Msg Post Rtn		
Log File Dir	GTS.LOG	
Log Detail Level	Full	
Offline Queue	☐	
Maint Msg Dets	✓	Y
Det Prefix	SEAT	
In Queue Dir		
In Queue Name		
Out Queue Dir		
Out Queue Name		
Queue Init Rtn		
Queue Close Rtn		
Syntax Type	Gts [✓ Ofs] Xml	
Local Ref		
Generic User	INPUTTER	
In Dir Rtn		
Version		
Ib User Check	[✓ [None]] N Y	
Eod Validate	☐	
Field Val	[None] No [✓ Yes]	
Attributes.1	TWS	
Attributes.2	PREAUTHENTICATED	
Attributes.3	INTERNET	
Same Authoriser	✓	YES
Channel	INTERNET	Internet
Pswd Encrypted	[✓ [None]] No Yes	

javascript:void(0)

2. Remove the TWS attribute in the OFS.Source



OFS.SOURCE TCIB (Model Bank)

Description	Connect Internet Banking	
Source Type	Telnet	
Login Id.1	tws	
Eb Phant Id.1		
Max Connections		
Restrict Link	[✓ [None]] Close Enq	
Initial Routine		
Close Routine		
In Msg Rtn		
Out Msg Rtn		
Msg Pre Rtn	TCIB.LAST.LOGIN.UPDATE	Last login update
Msg Post Rtn		
Log File Dir	GTS.LOG	
Log Detail Level	Full	
Offline Queue	☐	
Maint Msg Dets	✓	Y
Det Prefix	SEAT	
In Queue Dir		
In Queue Name		
Out Queue Dir		
Out Queue Name		
Queue Init Rtn		
Queue Close Rtn		
Syntax Type	Gts [✓ Ofs] Xml	
Local Ref		
Generic User	INPUTTER	
In Dir Rtn		
Version		
Ib User Check	[✓ [None]] N Y	
Eod Validate	☐	
Field Val	[None] No [✓ Yes]	
Attributes.1	TWS	
Attributes.2	PREAUTHENTICATED	
Attributes.3	INTERNET	
Same Authoriser	✓	YES
Channel	INTERNET	Internet
Pswd Encrypted	[✓ [None]] No Yes	
Ofs Message Decrypt		

javascript:void(0)

Note: Restart the T24 Application to apply the configuration for the application runtime.

API Dev

An API defined in Swagger 2.0 is imported into an API project. Import process will create the Camel routes, mock responses and wire the routes to the mock responses. This is done using a Maven plugin, either from your IDE or from the command line.

This module covers the following topics.

Getting Started

The table below describes the technical terminologies used in the solution.

Terms	Description
Swagger	Swagger is an open source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services.
Apache Camel	Apache Camel is an open source framework for message-oriented middleware with a rule-based routing and mediation engine that provides a Java object-based implementation of the Enterprise Integration Patterns using an application programming interface (or declarative Java domain-specific language) to configure routing and mediation rules.
Archetype	It is defined as an original pattern or model from which all other things of the same kind are made.
Maven	Maven is a build automation tool used primarily for Java projects.
Inventory	It is a file that provides the information of an API in core banking terms. It helps in creating the service implementation by generating swagger specification of an API.
payload	The actual information or message in transmitted data, as opposed to automatically generated metadata.
Deployment Descriptor	A deployment descriptor refers to a configuration file for an artifact that is deployed to some container.

The following steps are required for API development:

- Creation of Artefacts in T24- This section explains how data is retrieved from T24 system and provided to IRIS to expose to external requests. This in turn reduces the data loads being retrieved and simplifies the whole process. Also, it can be referred as filtering of T24 data by T24 being expose to the external request.
- Creation of Service Project- It contains the default directories and set of model data. It basically acts as a platform where developers can build and run the required APIs. It also contains a set of archetypes which gives developers options of creating desired APIs.
- Creation of Provider API- Provider APIs expose core banking capabilities as RESTful APIs. The key concept is that each Provider API is driven from an inventory that defines the contents of the API in core banking terms. The inventory is used to create the Swagger specification of the API. Together, the inventory file and the generated swagger specification are used to create the service implementation.
- Creation of Published API- Publisher APIs provide the following REST resources:

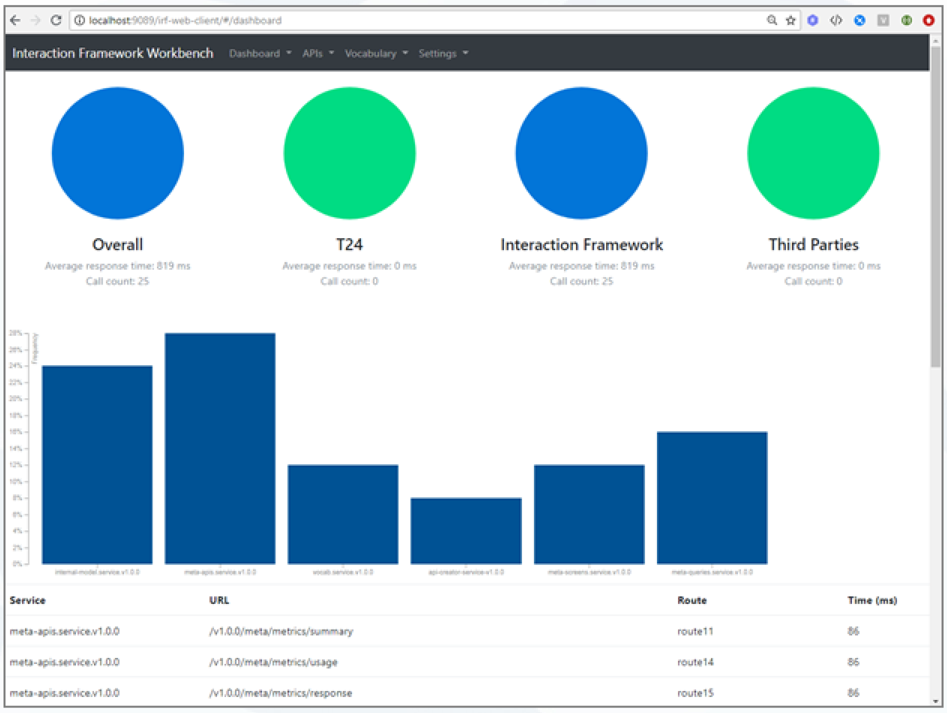
- Login
- Logout
- Add API, Update API, Remove API, Copy an API
- Validation roles

Note: When you access any API other than the login and logout APIs through an external REST client, first invoke the login API to ensure that user is authenticated. When the login API is invoked, the system stores the generated session cookie in a file, which we use in the next API invocations.

Workbench Installation

IRIS R18 Workbench

IRIS R18 Workbench is a Licensed Design-Time Tool that can be effectively used to create Provider APIs as well as Publisher APIs. A separate request is required to make to the Distribution to acquire the workbench which facilitates you to create the APIs on your own.



localhost:9089/irf-web-client/#/dashboard
Interaction Framework Workbench
Dashboard
APIs
Vocabulary
Settings

Overall
Average response time: 819 ms
Call count: 25

T24
Average response time: 0 ms
Call count: 0

Interaction Framework
Average response time: 819 ms
Call count: 25

Third Parties
Average response time: 0 ms
Call count: 0

Frequency
internal-model.service.v1.0.0
meta-apis.service.v1.0.0
vocab.service.v1.0.0
api-creator-service.v1.0.0
meta-screens.service.v1.0.0
meta-queries.service.v1.0.0

Service	URL	Route	Time (ms)
meta-apis.service.v1.0.0	/v1.0.0/meta/metrics/summary	route11	86
meta-apis.service.v1.0.0	/v1.0.0/meta/metrics/usage	route14	86
meta-apis.service.v1.0.0	/v1.0.0/meta/metrics/response	route15	86

IRIS Workbench Packaging

For acquiring the package, a request should be made for IRIS R18 design time package which will follow the naming convention IRISR18_Design_vxx.xx.xx.zip where xx denotes the release version.

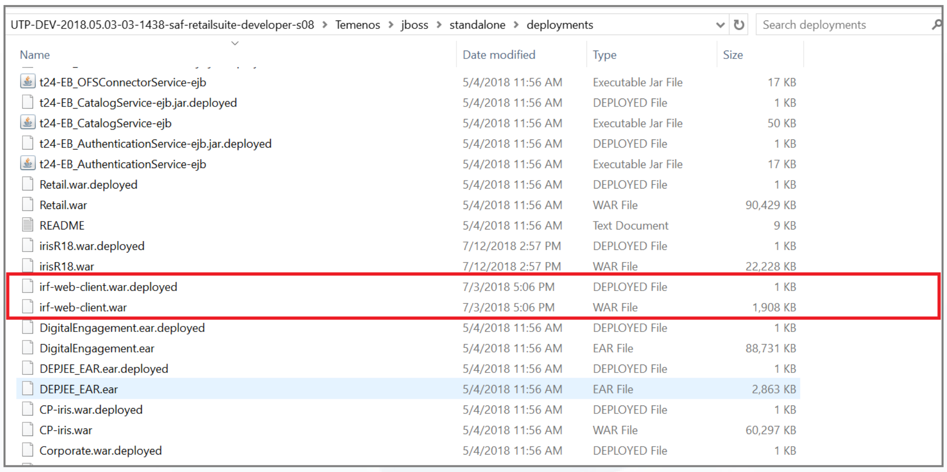
This package will mainly consist of the following components :

- irf-web-client.war - This is the Workbench war which needs to extracted from the zip package and can be deployed in any of the application servers(jboss, weblogic, websphere). It is required to connect to T24 via jetty server or default IRIS war.
- irf-test-web.war - This is an out of box pre-configured IRIS R18 war which will by default contain PSD2 APIs. This war can be deployed in jboss and can be used by the Workbench to connect to T24 during Design Time (steps mentioned below to configure the workbench for it).
- irf-repo - This is the IRIS R18 Framework artefacts that forms the repository and are needed while generating an IRIS war from the container project.

IRIS Workbench Configurations Steps

Step 1 : Extract the irf-web-client.war from the received Design time package IRISR18_Design_vxx.xx.xx.zip.

Step 2 : Deploy the irf-web-client.war in deployment folder of Jboss Application Server.

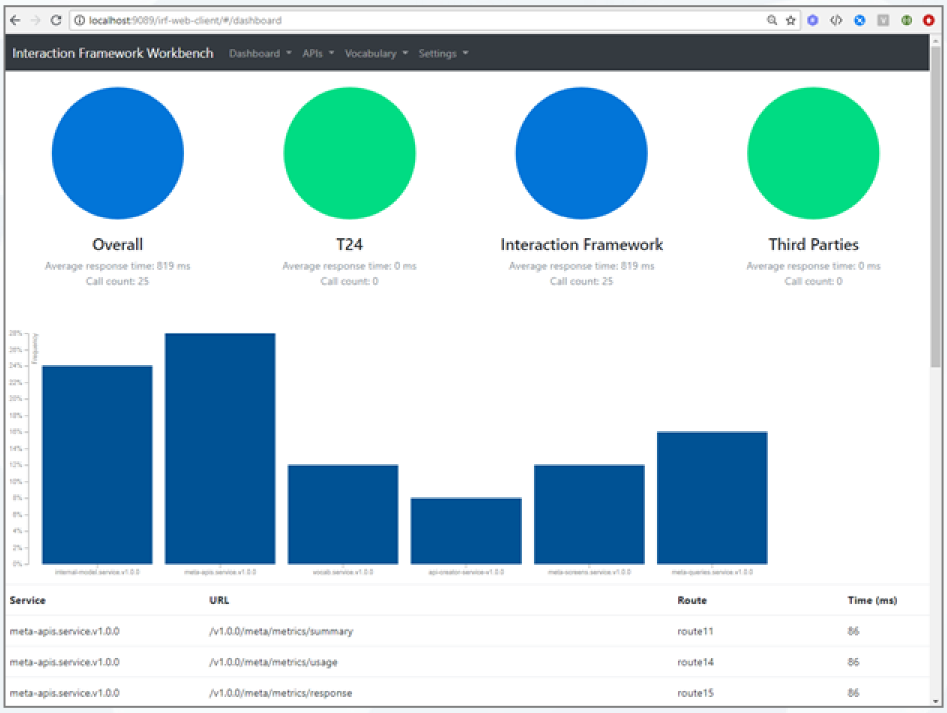


UTP-DEV-2018.05.03-03-1438-saf-retailsuite-developer-s08 › Temenos › jboss › standalone › deployments

Name	Date modified	Type	Size
t24-EB_OFSConnectorService-ejb	5/4/2018 11:56 AM	Executable Jar File	17 KB
t24-EB_CatalogService-ejb.jar.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB
t24-EB_CatalogService-ejb	5/4/2018 11:56 AM	Executable Jar File	50 KB
t24-EB_AuthenticationService-ejb.jar.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB
t24-EB_AuthenticationService-ejb	5/4/2018 11:56 AM	Executable Jar File	17 KB
Retail.war.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB
Retail.war	5/4/2018 11:56 AM	WAR File	90,429 KB
README	5/4/2018 11:56 AM	Text Document	9 KB
irisR18.war.deployed	7/12/2018 2:57 PM	DEPLOYED File	1 KB
irisR18.war	7/12/2018 2:57 PM	WAR File	22,228 KB
irf-web-client.war.deployed	7/3/2018 5:06 PM	DEPLOYED File	1 KB
irf-web-client.war	7/3/2018 5:06 PM	WAR File	1,908 KB
DigitalEngagement.ear.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB
DigitalEngagement.ear	5/4/2018 11:56 AM	EAR File	88,731 KB
DEPJEE_EAR.ear.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB
DEPJEE_EAR.ear	5/4/2018 11:56 AM	EAR File	2,863 KB
CP-iris.war.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB
CP-iris.war	5/4/2018 11:56 AM	WAR File	60,297 KB
Corporate.war.deployed	5/4/2018 11:56 AM	DEPLOYED File	1 KB

Step 3 : Verify that the irf-web-client.war is deployed successfully and Workbench is up and running.

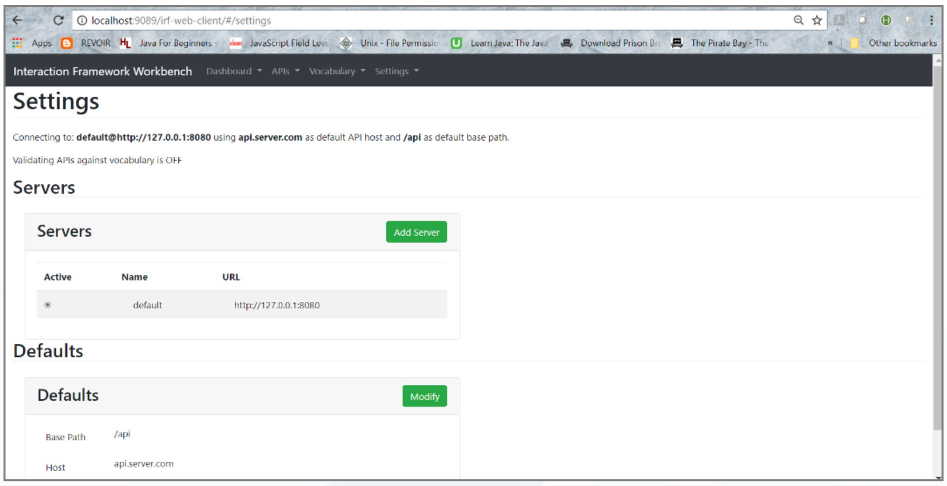
- Open New Browser Window & type http://localhost:9089/irf-web-client/


localhost:9089/irf-web-client/#/dashboard
Interaction Framework Workbench
Dashboard
APIs
Vocabulary
Settings
Overall
Average response time: 819 ms
Call count: 25
T24
Average response time: 0 ms
Call count: 0
Interaction Framework
Average response time: 819 ms
Call count: 25
Third Parties
Average response time: 0 ms
Call count: 0
Frequency
internal-model.service.v1.0.0
meta-apis.service.v1.0.0
vocab.service.v1.0.0
api-creator-service.v1.0.0
meta-screens.service.v1.0.0
meta-queries.service.v1.0.0

Service	URL	Route	Time (ms)
meta-apis.service.v1.0.0	/v1.0.0/meta/metrics/summary	route11	86
meta-apis.service.v1.0.0	/v1.0.0/meta/metrics/usage	route14	86
meta-apis.service.v1.0.0	/v1.0.0/meta/metrics/response	route15	86

Step 4 : To start creating Provider APIs, you are required to connect to T24 to get a list of artefacts for which you intend to generate an API service. There are two ways to do this :

- First way is using jetty server started from the created container project. For that, ensure that the setting in the workbench has been properly configured and T24 is up and running. By Default , jetty runs on 8080 port.


localhost:9089/irf-web-client/#/settings
Interaction Framework Workbench
Dashboard
APIs
Vocabulary
Settings
Add Server
Modify

Settings

Connecting to: **default@http://127.0.0.1:8080** using **api.server.com** as default API host and **/api** as default base path.

Validating APIs against vocabulary is OFF

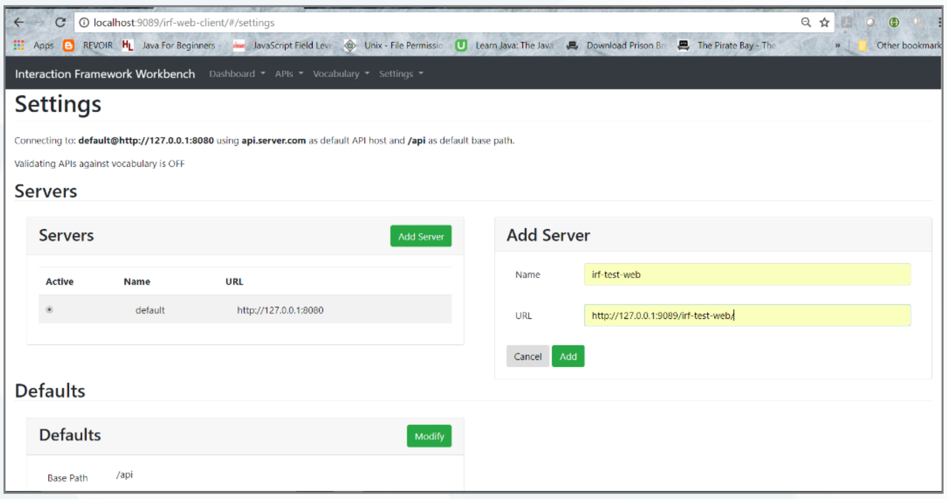
Servers

Active	Name	URL
●	default	http://127.0.0.1:8080

Defaults

Base Path	/api
Host	api.server.com

- Second way is to connect via irf-test-web.war provided as a part of the IRIS Design time package. For that you need to extract the irf-test-web.war from the package and deploy it in the jboss server. Once successfully deployed, you need to configure the WorkBench by adding a new server in the settings as shown below to connect to T24 via pre-configured irf-test-web.war


localhost:9089/irf-web-client/#/settings
Interaction Framework Workbench
Dashboard
APIs
Vocabulary
Settings
Add Server
Cancel
Add
Modify

Settings

Connecting to: **default@http://127.0.0.1:8080** using **api.server.com** as default API host and **/api** as default base path.

Validating APIs against vocabulary is OFF

Servers

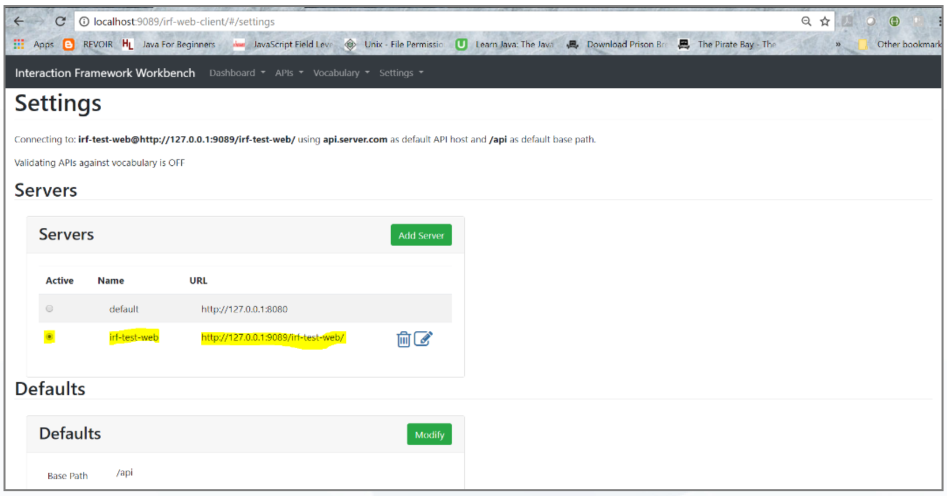
Active	Name	URL
●	default	http://127.0.0.1:8080

Add Server

Name	irf-test-web
URL	http://127.0.0.1:9089/irf-test-web/

Defaults

Base Path	/api



Settings

Connecting to: **irf-test-web@http://127.0.0.1:9089/irf-test-web/** using **api.server.com** as default API host and **/api** as default base path.

Validating APIs against vocabulary is OFF

Servers

Servers

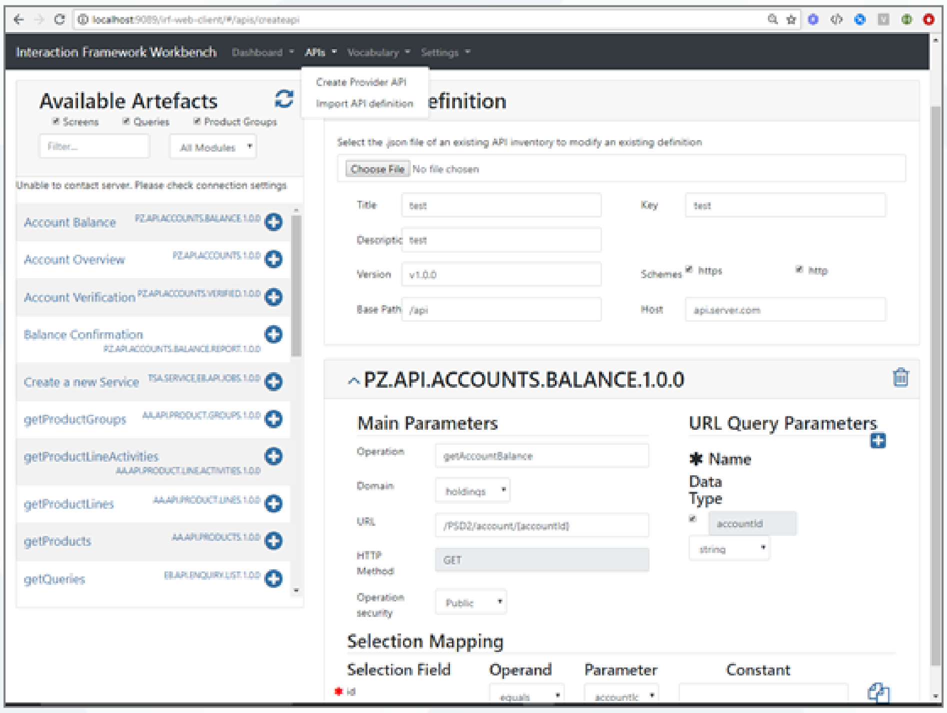
Active	Name	URL
○	default	http://127.0.0.1:8080
●	irf-test-web	http://127.0.0.1:9089/irf-test-web/

Defaults

Defaults

Base Path	/api

Step 5 : Once properly configured, you should hover to APIs → Create Provider API and the Workbench list you all the available artefacts in T24\. You are ready to go and create your own APIs.



Available Artefacts

Unable to contact server. Please check connection settings

Account Balance	PZ.API.ACCOUNTS.BALANCE.1.0.0
Account Overview	PZ.API.ACCOUNTS.1.0.0
Account Verification	PZ.API.ACCOUNTS.VERIFIED.1.0.0
Balance Confirmation	PZ.API.ACCOUNTS.BALANCE.REPORT.1.0.0
Create a new Service	TSA.SERVICE.EB.API.JOBS.1.0.0
getProductGroups	AA.API.PRODUCT.GROUPS.1.0.0
getProductLineActivities	AA.API.PRODUCT.LINE.ACTIVITIES.1.0.0
getProductLines	AA.API.PRODUCT.LINES.1.0.0
getProducts	AA.API.PRODUCTS.1.0.0
getQueries	EB.API.ENQUIRY.LIST.1.0.0

Select the .json file of an existing API inventory to modify an existing definition

No file chosen

Title	test	Key	test
Descriptic	test		
Version	v1.0.0	Schemes	☑ https ☑ http
Base Path	/api	Host	api.server.com

PZ.API.ACCOUNTS.BALANCE.1.0.0

Main Parameters

Operation	getAccountBalance
Domain	holdings
URL	/PSD2/account/{accountId}
HTTP Method	GET
Operation security	Public

URL Query Parameters

✱ Name	Data Type
☑ accountId	string

Selection Mapping

Selection Field	Operand	Parameter	Constant
id	equals	accountIc	

Creation of Artefacts in T24

In order for T24 Version and Enquiry artefacts to be used in the Interaction Framework, certain rules must be followed. These rules provide a simple and effective governance framework that

- Clearly identifies that a given T24 artefact is used in an API
- Allows simple versioning control to be applied to the T24 artefact following semantic versioning (refer http://semver.org/)
- Provides data type meta data to ensure clean conversion of data from T24 to internet standard data types
- Provides meaningful operation names

The following rules must be followed:

ID Naming Convention

For Enquiry, the following ID naming convention must be followed:

```
XX.API.COLLECTIONID.SUBCOLLECTIONID.Major.Minor.Patch
```

Examples: PZ.API.ACCOUNTS.BALANCES.1.0.0, PZ.API.ACCOUNTS.TRANSACTIONS.1.0.0

For Version the following ID naming convention must be followed:

```
TABLE, XX.API.VERB.RESOURCEID.Major.Minor.Patch
```

Example: FUNDS.TRANSFER, AC.API.CREATE.TRANSFER.1.0.0

Field data typing

Each Enquiry column and Version field must have a data type defined.

For Enquiry, this is the FIELD.DISP.TYPE field,

For VERSION the ATTRIBS field is used.

Fields are considered to be alphanumeric unless they are set as **Date** or **Amount**. Amount and date fields must be unformatted, e.g. 20170123 and 1234.56

S.No	T24 Data Type	Swagger Schema Data Type
1	ALPHANUMERIC	string
2	AMOUNT	number
3	DATE	Date
4	TIMESTAMP	Date-time
5	BOOLEAN	Boolean

Description

Each Enquiry and Version must have a description. For Enquiry, this is the DESCRIPT field, for Version the DESCRIPTION field is used.

This is used to describe the operation supported and should follow verb/resource/subresource, e.g. getAccountTransactions, getAccountBalances, createAccountTransfer

Restrictions on Enquiry

Header fields should not be used in API enquiries.

- Turn SEL.LABEL into API style labels
- remove HEADER
- Turn FIELD.LABEL into API style labels
- Use FIELD.DISPLAY.TYPE for data type (Alphanumeric / amount, date)
- remove any static labels - where OPERATION is a literal and COLUMN is set
- Set ATTRIBS to ''
- Remove drill down information (ENQUIRY.NAME,SEL.CRIT,LABEL.FIELD,NXT.DESC)
- Set DESCRIPT to be the operation, e.g. getAccountBalances

Naming should be in standard vocabulary

The standard Temenos vocabulary exists to bring consistent naming across all API initiatives. Field / main resource / sub resource names should exist in the standard vocabulary.

Click here to know **how the T24 enquiry definition affects the JSON response**.

> Note: If there is an existing Service Project, you can skip Create Service Project step and navigate to Provider API step.

Creation of Service Container

In order to run any API projects, you should have a service container. A service container hosts one or more service projects. It is also a runtime container that hosts the camel run time and configuration required to connect to downstream systems, i.e. T24

You can create the service container using the supplied Maven Archetype, either from your IDE or from the command line.

Using the command line

mvn archetype:generate -DarchetypeGroupId=com.temenos.irf -DarchetypeArtifactId=irf-service-container-archetype

-DarchetypeVersion=1.0

Supply values for various arguments, either interactively or as parameters from the command line:

Define value for property 'groupId': com.temenos.training.irf

Define value for property 'artifactId': training-container

Define value for property 'version' 1.0-SNAPSHOT: :

Define value for property 'package' com.temenos.training.irf::

Confirm properties configuration:

groupId: com.temenos.training.irf

artifactId: training-container

version: 1.0.0-SNAPSHOT

package: com.temenos.training.irf

As a single command:

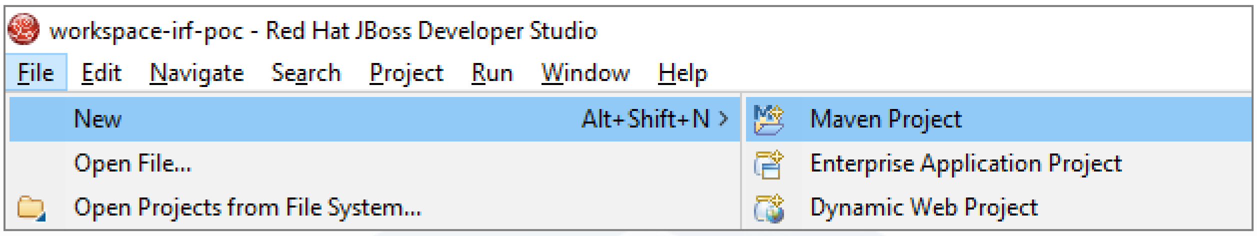
mvn archetype:generate -DarchetypeGroupId=com.temenos.irf -DarchetypeArtifactId=irf-service-container-archetype

-DarchetypeVersion=1.0 -DartifactId=training-container

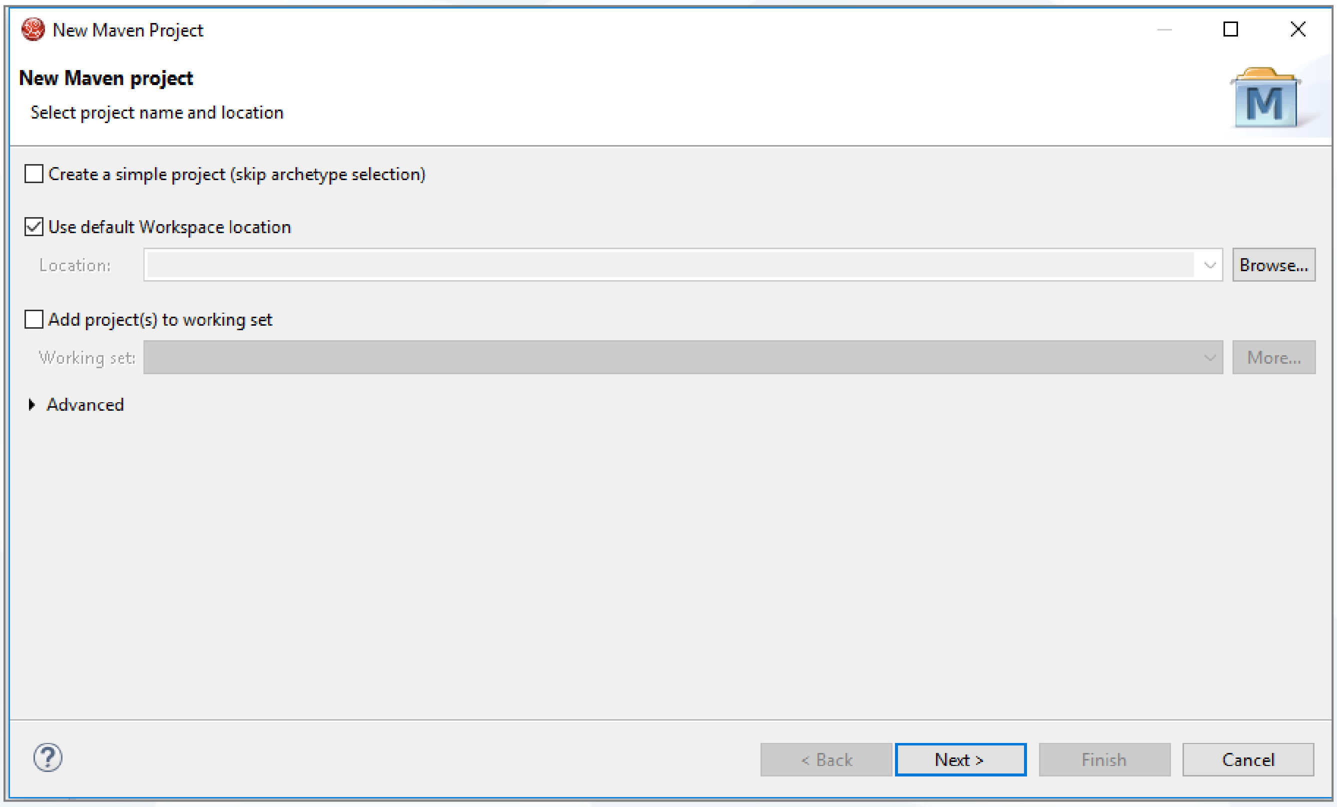
-DgroupId=com.temenos.irf.training -Dpackage=com.temenos.irf.training -Dversion=SNAPSHOT-0.0.1

Using Eclipse

From the File menu, select File > New > Maven Project


workspace-irf-poc - Red Hat JBoss Developer Studio
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Maven Project
Enterprise Application Project
Dynamic Web Project

Click the Next button.



New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☑ Use default Workspace location

Location:

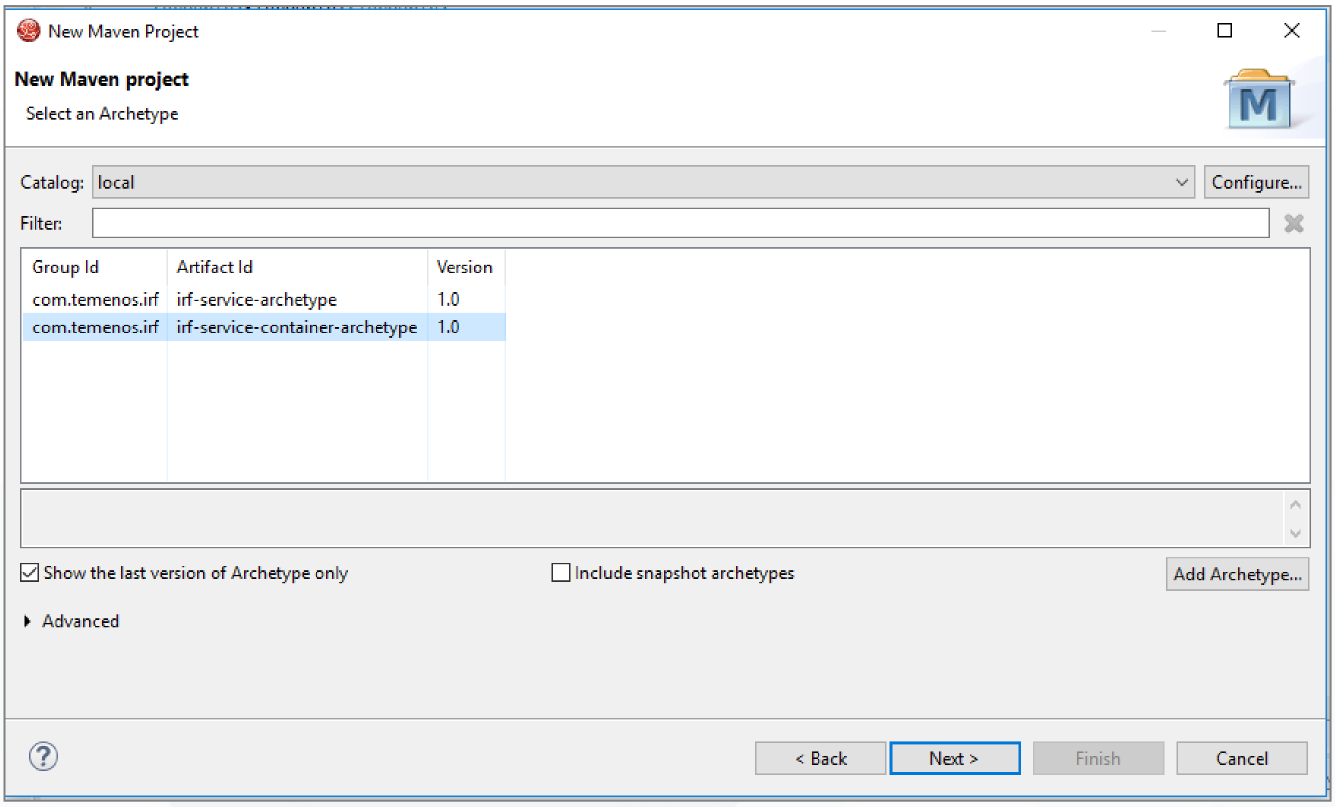
☐ Add project(s) to working set

Working set:

Advanced

Select Archetype

Select the **irf-service-container-archetype** option from Artifact Id and click the **Next** button.


New Maven Project

New Maven project

Select an Archetype

Catalog: local

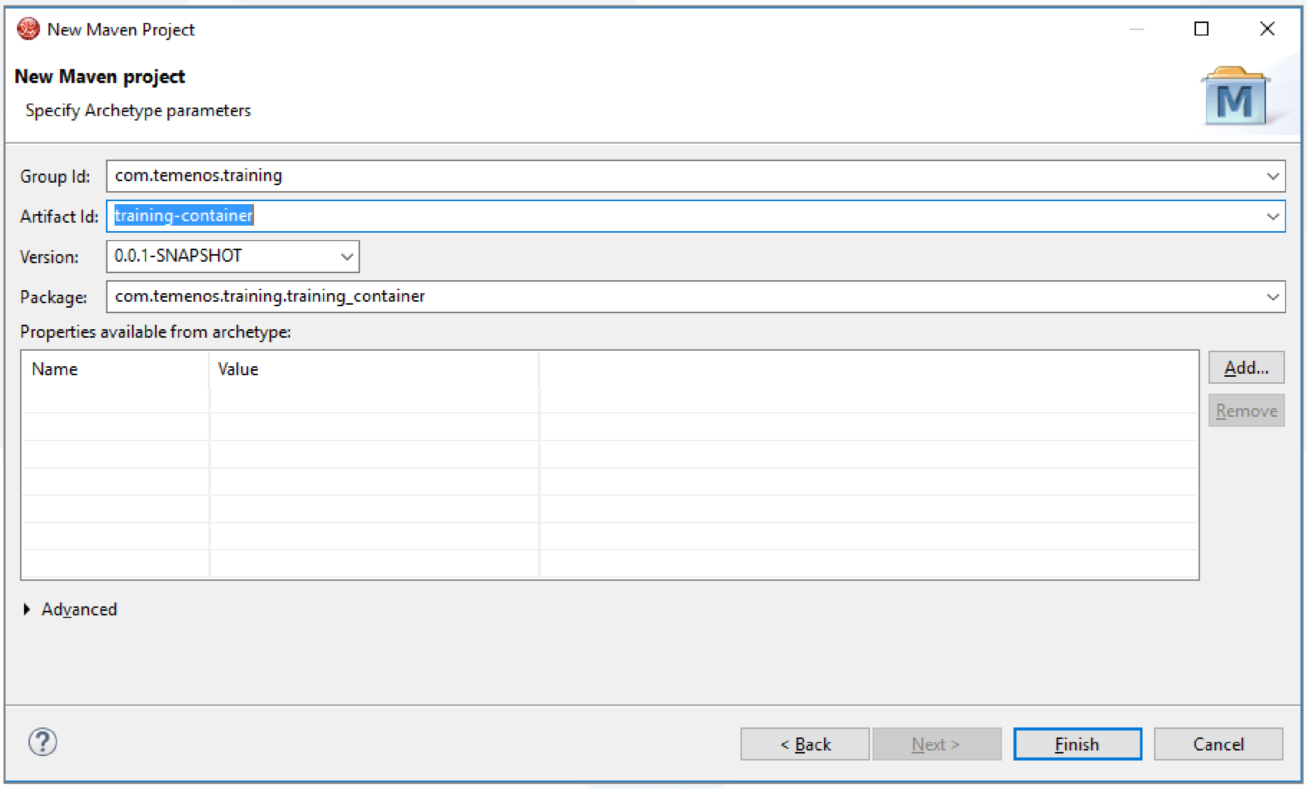
Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	1.0
com.temenos.irf	irf-service-container-archetype	1.0

☑ Show the last version of Archetype only ☐ Include snapshot archetypes

Provide Project Details

Fill in the **Group Id**, **Artifact Id** and **Version** fields, and click the **Finish** button.


New Maven Project

New Maven project

Specify Archetype parameters

Group Id: com.temenos.training

Artifact Id: training-container

Version: 0.0.1-SNAPSHOT

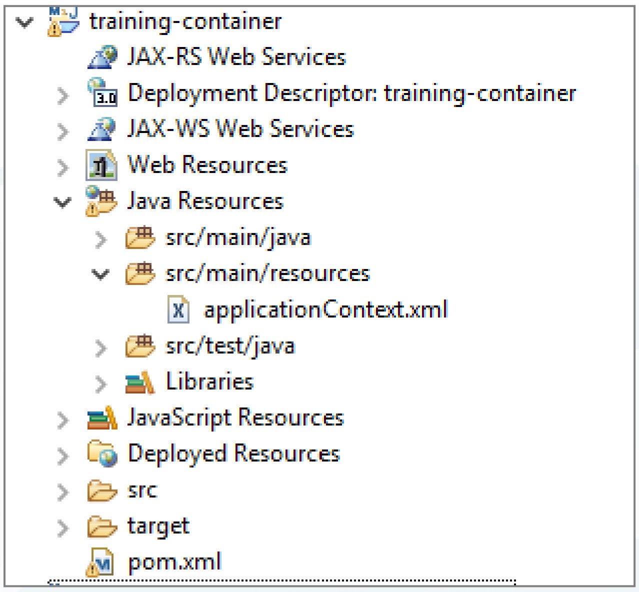
Package: com.temenos.training.training_container

Properties available from archetype:

Name	Value

Project Structure

You should see a project created with the following structure:



```
training-container
  JAX-RS Web Services
  Deployment Descriptor: training-container
  JAX-WS Web Services
  Web Resources
  Java Resources
    src/main/java
    src/main/resources
      applicationContext.xml
    src/test/java
    Libraries
  JavaScript Resources
  Deployed Resources
  src
  target
  pom.xml
```

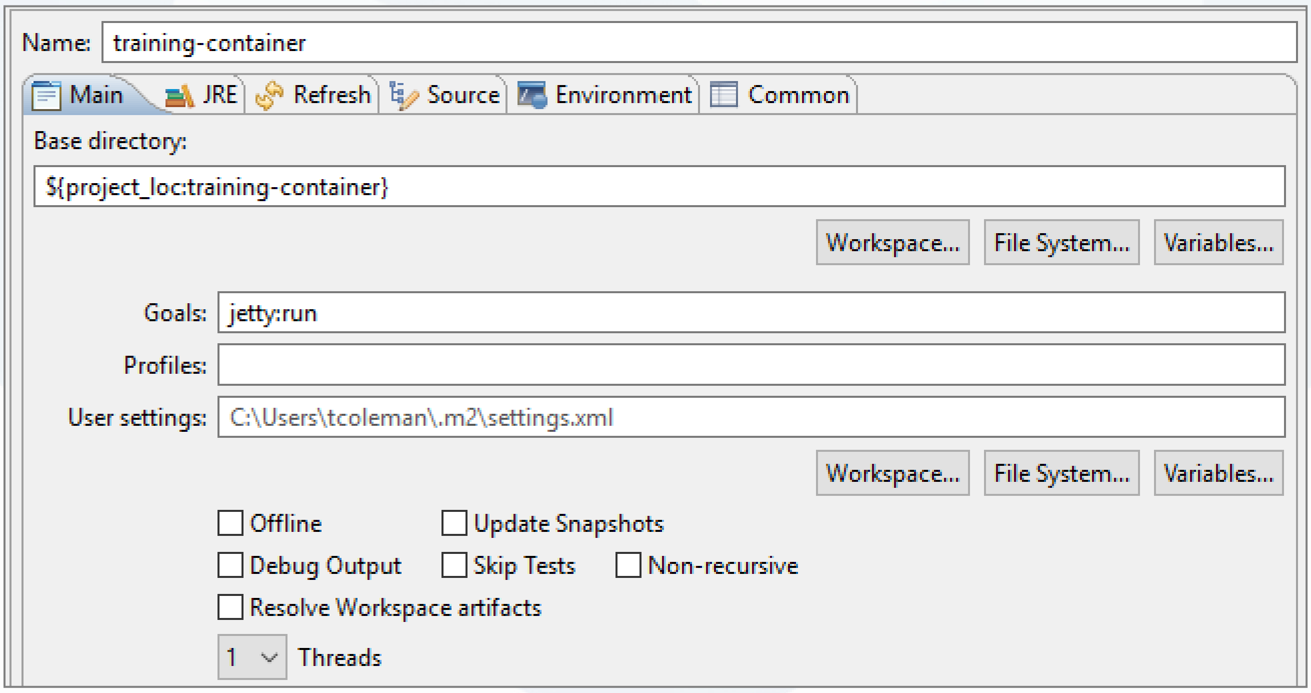
Start the server

You can now start the server, either from the command line or from your IDE using mvn jetty:run

mvn jetty:run

In Eclipse, right click the project and select Run As > Maven Build

In the dialog box, set the Goals to jetty:run



Name: training-container

Base directory: ${project_loc:training-container}

Goals: jetty:run

Profiles:

User settings: C:\Users\tcoleman\.m2\settings.xml

☐ Offline ☐ Update Snapshots

☐ Debug Output ☐ Skip Tests ☐ Non-recursive

☐ Resolve Workspace artifacts

1 Threads

Creation of Service Project

Create a service project using the irf-service-archetype.

Using the command line

mvn archetype:generate -DarchetypeGroupId=com.temenos.irf -DarchetypeArtifactId=irf-service-archetype -DarchetypeVersion=1.0 -DarchetypeCatalog=local

Supply values for various arguments, either interactively or as parameters from the command line:

Define value for property 'groupId': com.temenos.training.irf

Define value for property 'artifactId': training-api

Define value for property 'version' 1.0-SNAPSHOT: :

Define value for property 'package' com.temenos.training.irf::

Confirm properties configuration:

groupId: com.temenos.training.irf

artifactId: training-api

version: 1.0-SNAPSHOT

package: com.temenos.training.irf

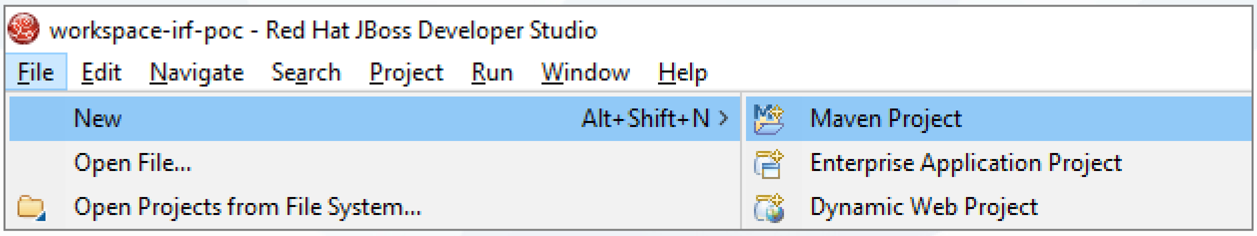
As a single command:

mvn archetype:generate -DarchetypeGroupId=com.temenos.irf -DarchetypeArtifactId=irf-service-archetype -DarchetypeVersion=1.0 -DartifactId=training-api

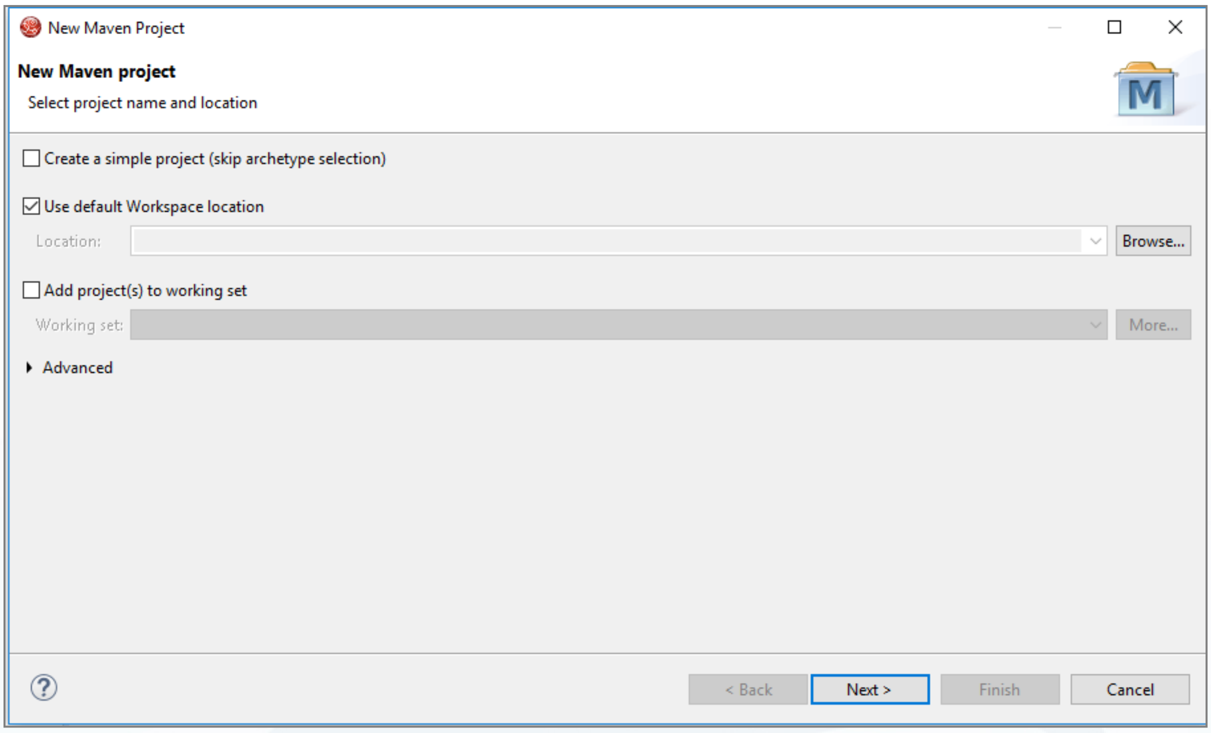
-DgroupId=com.temenos.irf.training -Dpackage=com.temenos.irf.training -Dversion=SNAPSHOT-0.0.1

Using Eclipse

From the File menu, select **File** > **New** > **Maven Project**


workspace-irf-poc - Red Hat JBoss Developer Studio
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Maven Project
Enterprise Application Project
Dynamic Web Project

Click the **Next** button.


New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☑ Use default Workspace location

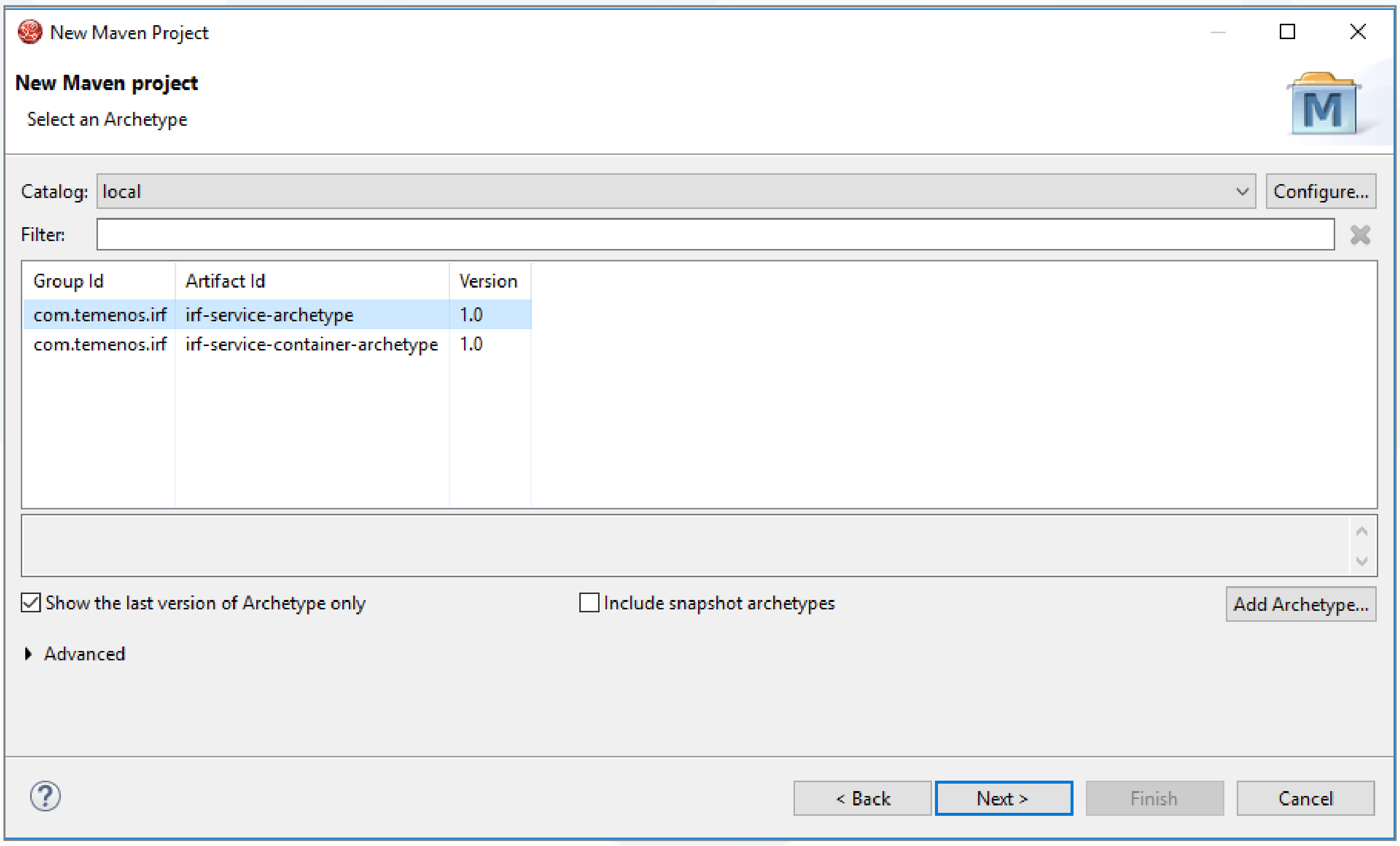
Location:

☐ Add project(s) to working set

Working set:

Select archetype

Select the **irf-service-archetype** option from Artifact Id column and click the **Next** button.


New Maven Project

New Maven project

Select an Archetype

Catalog: local

Filter:

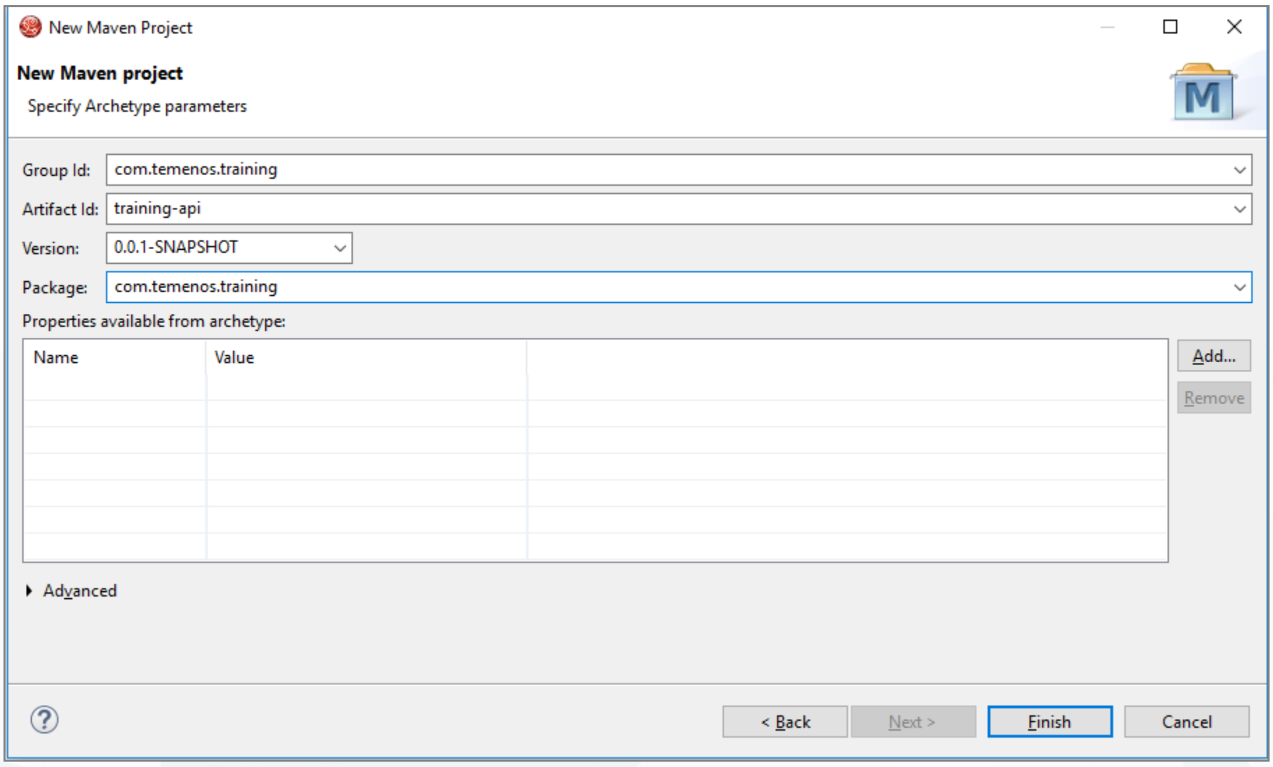
Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	1.0
com.temenos.irf	irf-service-container-archetype	1.0

☑ Show the last version of Archetype only

☐ Include snapshot archetypes

Provide Project details

Fill in the **Group Id**, **Artifact Id** and **Version** fields, and click the **Finish** button.



New Maven Project

New Maven project

Specify Archetype parameters

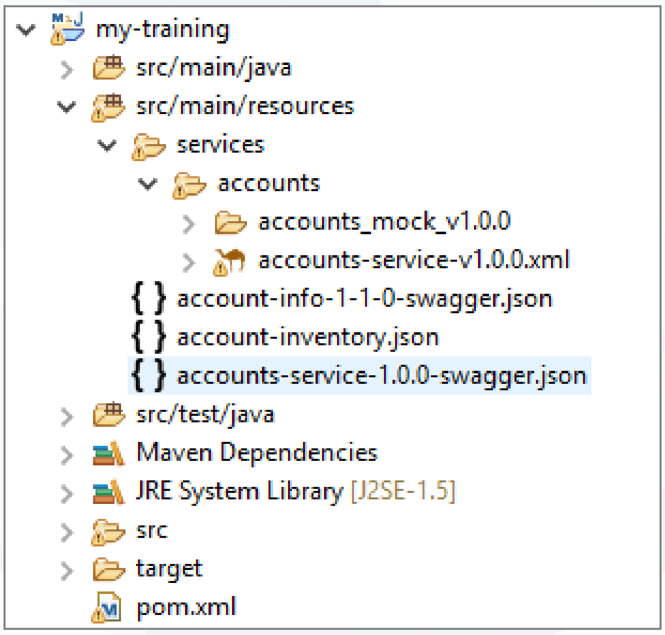
Group Id:	com.temenos.training
Artifact Id:	training-api
Version:	0.0.1-SNAPSHOT
Package:	com.temenos.training

Properties available from archetype:

Name	Value

Project Structure

You should see a project created with the following structure:



```
my-training
  src/main/java
  src/main/resources
    services
      accounts
        accounts_mock_v1.0.0
        accounts-service-v1.0.0.xml
    account-info-1-1-0-swagger.json
    account-inventory.json
    accounts-service-1.0.0-swagger.json
  src/test/java
  Maven Dependencies
  JRE System Library [J2SE-1.5]
  src
  target
  pom.xml
```

Using Workbench to Define Provider API

The easiest mechanism to create a Provider API is to use the Interaction Framework Workbench. Select "APIs > Create Provider API":

Step 1 - Start Service Container

Make sure that, Service Container is up and running.

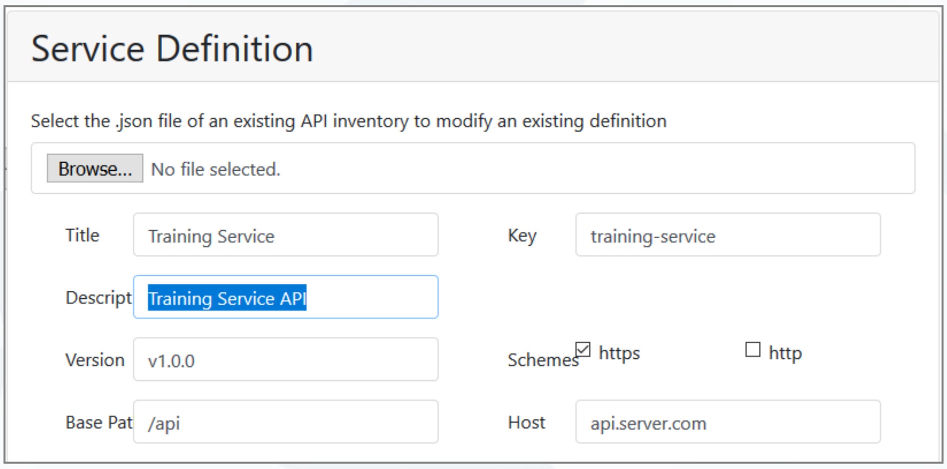
http://localhost:8080/api/v1.0.0/meta/apis

If it is not running, start the server using the following command

```
    jetty:run    from maven command prompt
```

Step 2 - Service Definition

Provide the Title, service Key, Description , etc in the Service Definition section



Service Definition

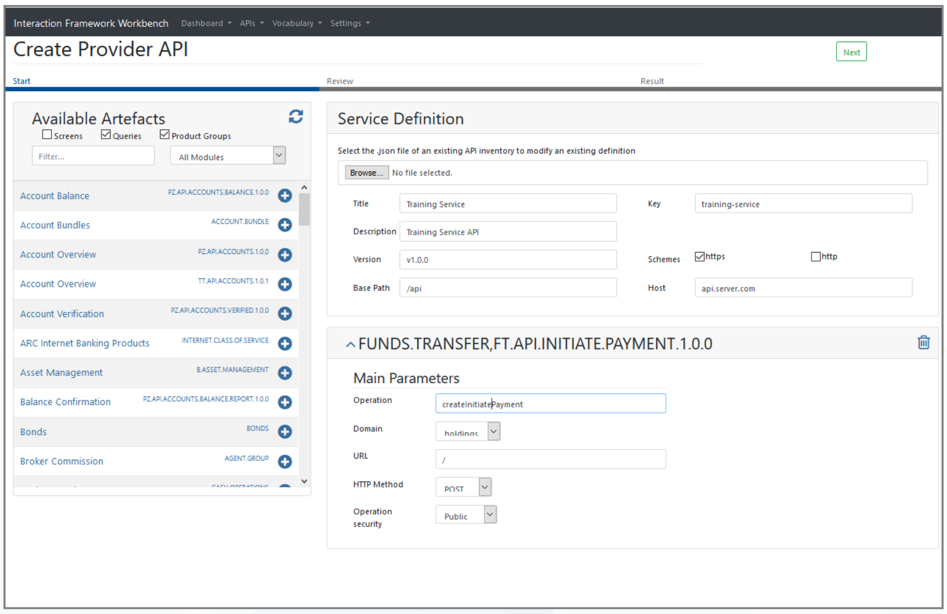
Select the .json file of an existing API inventory to modify an existing definition

Browse... No file selected.

Title	Training Service	Key	training-service
Descript	Training Service API		
Version	v1.0.0	Schemes	☑ https ☐ http
Base Pat	/api	Host	api.server.com

Step 3 - Select Artefacts to include

Artefacts may be filtered to aid in discovery. Clicking on an artefact will add an endpoint to the service, and create a URL based on the artefact definitions. Review the URL and Operation to ensure that this matches the API design. An end point may be removed by clicking on the trash icon in the top right of the endpoint definition.



Interaction Framework Workbench · Dashboard · APIs · Vocabulary · Settings

Create Provider API — Next

Start · Review · Result

Available Artefacts — ☐ Screens ☑ Queries ☑ Product Groups — Filter... — All Modules

Account Balance	PZ.API.ACCOUNTS.BALANCE.1.0.0
Account Bundles	ACCOUNT.BUNDLE
Account Overview	PZ.API.ACCOUNTS.1.0.0
Account Overview	TT.API.ACCOUNTS.1.0.1
Account Verification	PZ.API.ACCOUNTS.VERIFIED.1.0.0
ARC Internet Banking Products	INTERNET.CLASS.OF.SERVICE
Asset Management	B.ASSET.MANAGEMENT
Balance Confirmation	PZ.API.ACCOUNTS.BALANCE.REPORT.1.0.0
Bonds	BONDS
Broker Commission	AGENT.GROUP

Service Definition

Select the .json file of an existing API inventory to modify an existing definition

Browse... No file selected.

Title	Training Service	Key	training-service
Description	Training Service API		
Version	v1.0.0	Schemes	☑ https ☐ http
Base Path	/api	Host	api.server.com

FUNDS.TRANSFER,FT.API.INITIATE.PAYMENT.1.0.0

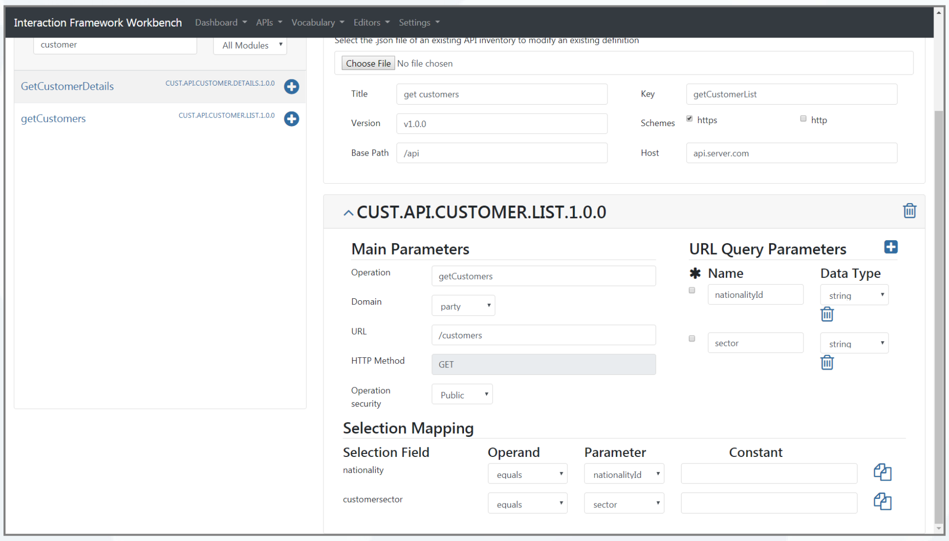
Main Parameters

Operation	createInitiatePayment
Domain	holdings
URL	/
HTTP Method	POST
Operation security	Public

Step 4 - Map URL parameters to Query Selection

Where a T24 VERSION is selected, only the method is required to be set. The payload will be configured in Swagger directly from the T24 Model. POST should be used to create new resources, PUT to amend and existing resource, GET for viewing a resources and DELETE for removing a resource.

Where a T24 ENQUIRY is selected, the URL parameters need to be mapped to the ENQUIRY selection fields.



Interaction Framework Workbench · Dashboard · APIs · Vocabulary · Editors · Settings

customer — All Modules

GetCustomerDetails	CUST.API.CUSTOMER.DETAILS.1.0.0
getCustomers	CUST.API.CUSTOMER.LIST.1.0.0

Select the .json file of an existing API inventory to modify an existing definition

Choose File No file chosen

Title	get customers	Key	getCustomerList
Version	v1.0.0	Schemes	☑ https ☐ http
Base Path	/api	Host	api.server.com

CUST.API.CUSTOMER.LIST.1.0.0

Main Parameters

Operation	getCustomers
Domain	party
URL	/customers
HTTP Method	GET
Operation security	Public

URL Query Parameters

Name	Data Type
nationalityId	string
sector	string

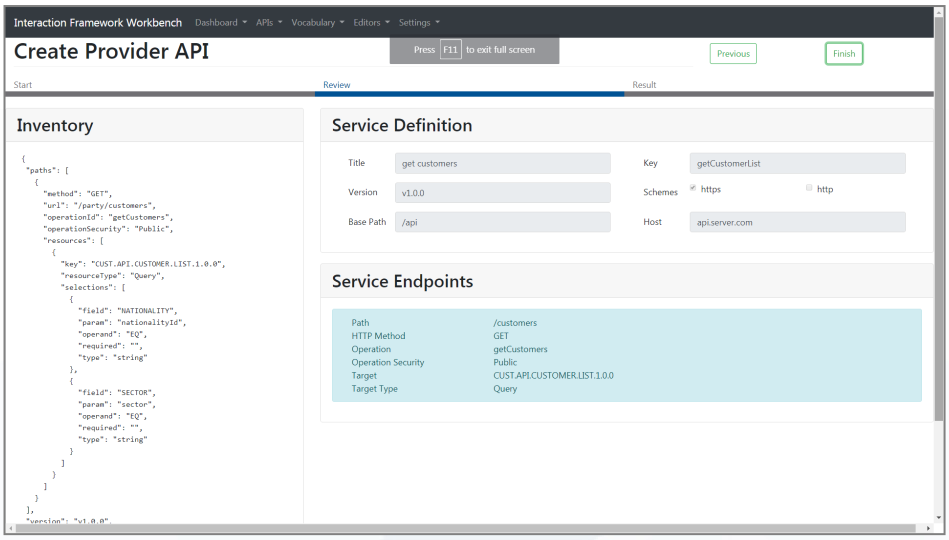
Selection Mapping

Selection Field	Operand	Parameter	Constant
nationality	equals	nationalityId	
customersector	equals	sector	

A selection field can be mapped from mulitple URL query or path parameters. In the above example, fromDate and toDate are both mapped to the valueDate selection field.

Click Next in the top right fo the screen to review the provider API

Step 5 - Review



Create Provider API

Inventory

```
{
 "paths": [
   {
     "method": "GET",
     "url": "/party/customers",
     "operationId": "getCustomers",
     "operationSecurity": "Public",
     "resources": [
       {
         "key": "CUST.API.CUSTOMER.LIST.1.0.0",
         "resourceType": "Query",
         "selections": [
           {
             "field": "NATIONALITY",
             "param": "nationalityId",
             "operand": "EQ",
             "required": "",
             "type": "string"
           },
           {
             "field": "SECTOR",
             "param": "sector",
             "operand": "EQ",
             "required": "",
             "type": "string"
           }
         ]
       }
     ]
   }
 ],
 "version": "v1.0.0",
```

Service Definition

Title	get customers	Key	getCustomerList
Version	v1.0.0	Schemes	✓ https ☐ http
Base Path	/api	Host	api.server.com

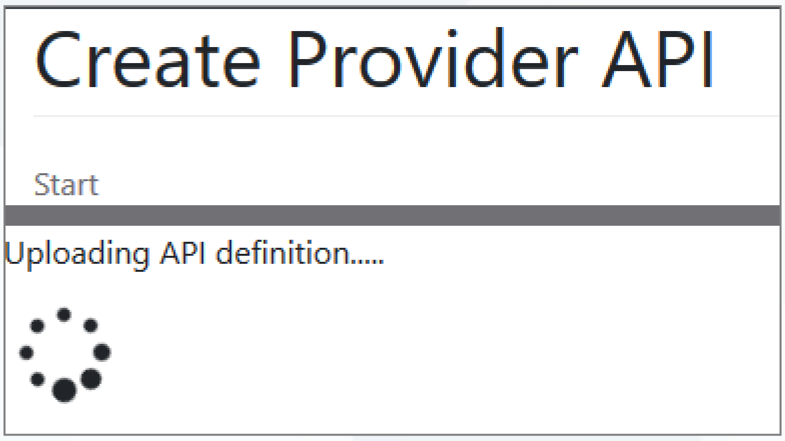
Service Endpoints

Path	/customers
HTTP Method	GET
Operation	getCustomers
Operation Security	Public
Target	CUST.API.CUSTOMER.LIST.1.0.0
Target Type	Query

The inventory file can be copied directly from the Inventory pane and pasted into an IDE. Or if preferred click Finish to create a zip file file containing the swager spec, Camel routes, mappings and mock responses.

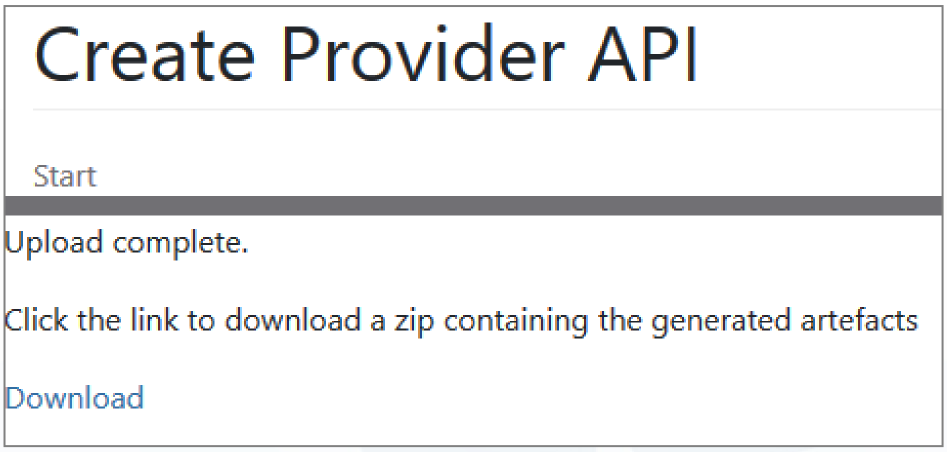
Step 6 (optional) - Submit

Click Finish the top right fo the screen to submit the provider API



Uploading API definition.....

The result provides a link to download a zip file containing the swager spec, Camel routes, mappings and mock responses.


Create Provider API

Start

Upload complete.

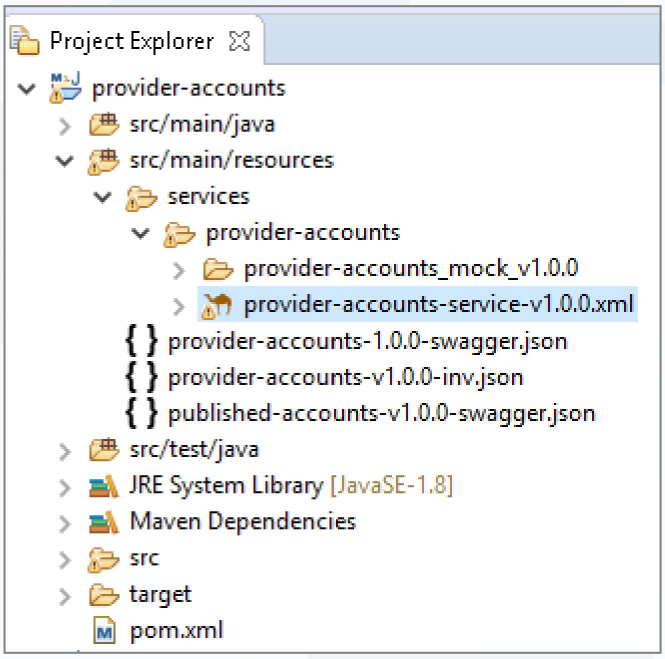
Click the link to download a zip containing the generated artefacts

Download

Step 7 Importing generated Artifacts

Extract the content of downloaded zip into <service-provider> project/ resources folder.

Project Structure



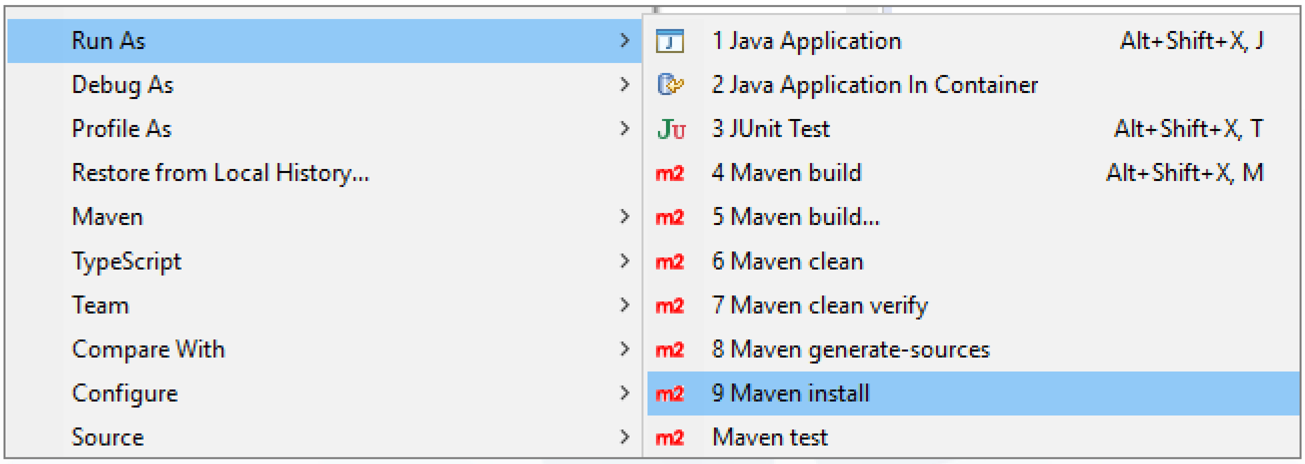
```
provider-accounts
  src/main/java
  src/main/resources
    services
      provider-accounts
        provider-accounts_mock_v1.0.0
        provider-accounts-service-v1.0.0.xml
      provider-accounts-1.0.0-swagger.json
      provider-accounts-v1.0.0-inv.json
      published-accounts-v1.0.0-swagger.json
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
```

Step 8 - Build The Provider API Project

Run the project as Maven > Install to build the project

Build the Service Project

Right click the project and choose Run As > Maven install to build the project.


Run As
Debug As
Profile As
Restore from Local History...
Maven
TypeScript
Team
Compare With
Configure
Source
1 Java Application Alt+Shift+X, J
2 Java Application In Container
3 JUnit Test Alt+Shift+X, T
4 Maven build Alt+Shift+X, M
5 Maven build...
6 Maven clean
7 Maven clean verify
8 Maven generate-sources
9 Maven install
Maven test

Creation of Provider API

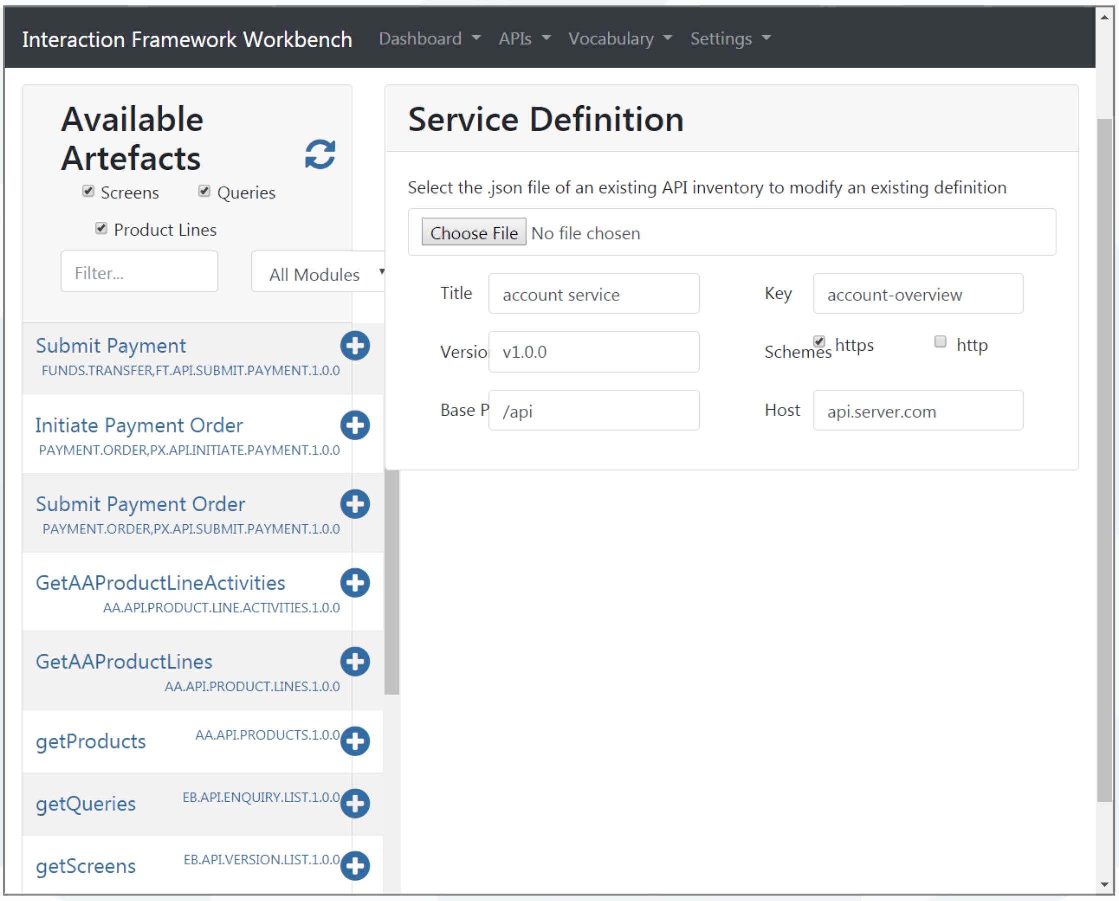
Provider APIs expose core-banking capabilities as RESTful APIs. The key concept is that each Provider API is driven from an **inventory** that defines the contents of the API in core banking terms. The inventory is used to create the Swagger specification of the API. Together, the inventory file and the generated swagger specification are used to create the service implementation.

Creating an Inventory using Interaction Framework Workbench

The easiest mechanism to create a Provider API is to use the Interaction Framework Workbench. Select "Create Provider API":

Step 1 - Service Definition

Provide the Group and Key (used in as the Group in the Maven artefact), the service title and the version of the service.



Service Definition

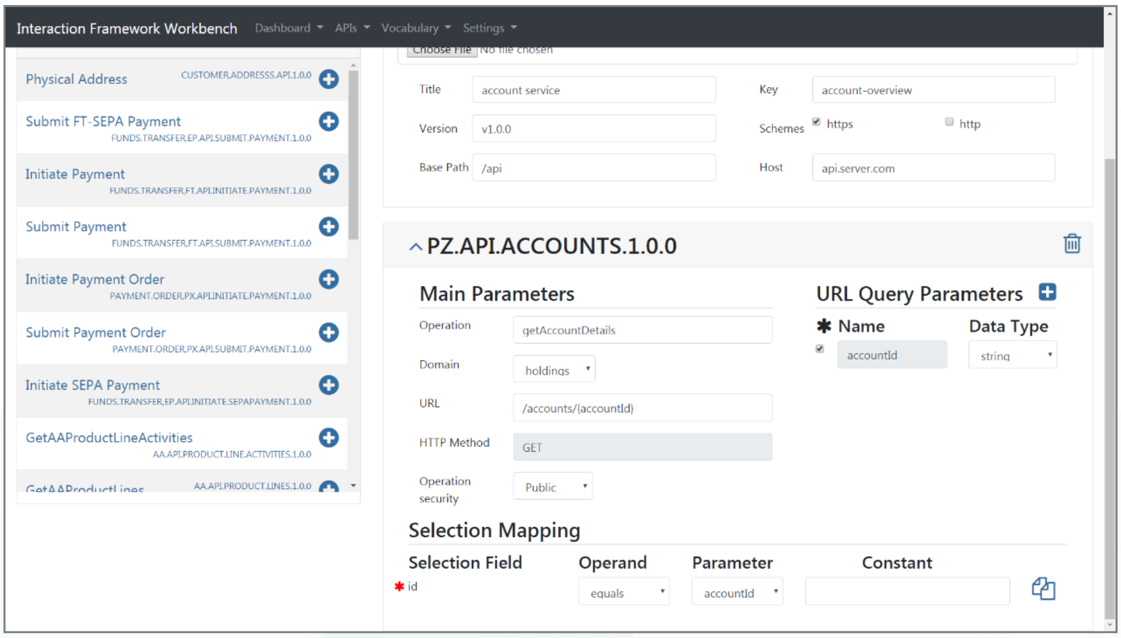
Select the .json file of an existing API inventory to modify an existing definition

Title	account service	Key	account-overview
Versio	v1.0.0	Schemes	[✓ https] http
Base P	/api	Host	api.server.com

Submit Payment	FUNDS.TRANSFER,FT.API.SUBMIT.PAYMENT.1.0.0
Initiate Payment Order	PAYMENT.ORDER,PX.API.INITIATE.PAYMENT.1.0.0
Submit Payment Order	PAYMENT.ORDER,PX.API.SUBMIT.PAYMENT.1.0.0
GetAAProductLineActivities	AA.API.PRODUCT.LINE.ACTIVITIES.1.0.0
GetAAProductLines	AA.API.PRODUCT.LINES.1.0.0
getProducts	AA.API.PRODUCTS.1.0.0
getQueries	EB.API.ENQUIRY.LIST.1.0.0
getScreens	EB.API.VERSION.LIST.1.0.0

Step 2 - Select Artefacts to include

Artefacts may be filtered to aid in discovery. Clicking on an artefact will add an endpoint to the service, and create a URL based on the artefact definitions. Review the URL and Operation to ensure that this matches the API design. An end point may be removed by clicking on the trash icon in the top right of the endpoint definition.



Physical Address	CUSTOMER,ADDRESSS.API.1.0.0
Submit FT-SEPA Payment	FUNDS.TRANSFER,EP.API.SUBMIT.PAYMENT.1.0.0
Initiate Payment	FUNDS.TRANSFER,FT.API.INITIATE.PAYMENT.1.0.0
Submit Payment	FUNDS.TRANSFER,FT.API.SUBMIT.PAYMENT.1.0.0
Initiate Payment Order	PAYMENT.ORDER,PX.API.INITIATE.PAYMENT.1.0.0
Submit Payment Order	PAYMENT.ORDER,PX.API.SUBMIT.PAYMENT.1.0.0
Initiate SEPA Payment	FUNDS.TRANSFER,EP.API.INITIATE.SEPAPAYMENT.1.0.0
GetAAProductLineActivities	AA.API.PRODUCT.LINE.ACTIVITIES.1.0.0
GetAAProductLines	AA.API.PRODUCT.LINES.1.0.0

Title: account service
Key: account-overview
Version: v1.0.0
Schemes: ✓ https ☐ http
Base Path: /api
Host: api.server.com

PZ.API.ACCOUNTS.1.0.0

Main Parameters

Operation: getAccountDetails
Domain: holdings
URL: /accounts/{accountId}
HTTP Method: GET
Operation security: Public

URL Query Parameters

✱ Name	Data Type
✓ accountId	string

Selection Mapping

Selection Field	Operand	Parameter	Constant
* id	equals	accountId	

Step 3 - Map URL parameters to Query Selection

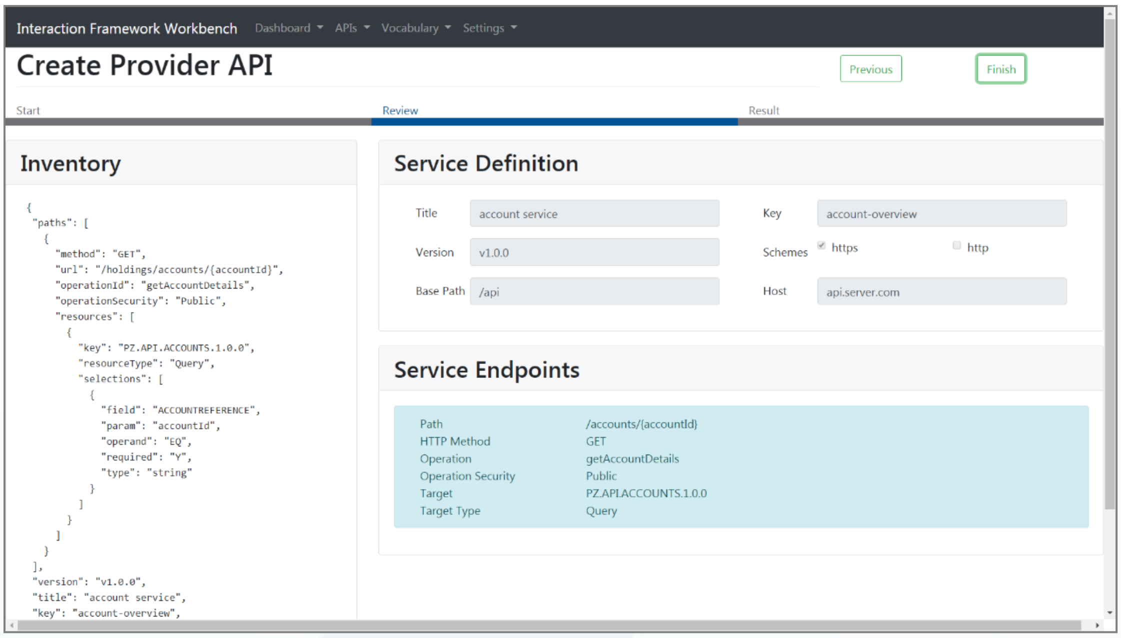
Where a T24 VERSION is selected, only the method is required to be set. The payload will be configured in Swagger directly from the T24 Model. POST should be used to create new resources, PUT to amend an existing resource, GET for viewing a resource and DELETE for removing a resource.

Where a T24 ENQUIRY is selected, the URL parameters need to be mapped to the ENQUIRY selection fields.

A selection field can be mapped from mulitple URL query or path parameters. In the above example, fromDate and toDate are both mapped to the valueDate selection field.

Click Next in the top right of the screen to review the Provider API.

Step 4 – Review



Create Provider API

Inventory

```
{
 "paths": [
   {
     "method": "GET",
     "url": "/holdings/accounts/{accountId}",
     "operationId": "getAccountDetails",
     "operationSecurity": "Public",
     "resources": [
       {
         "key": "PZ.API.ACCOUNTS.1.0.0",
         "resourceType": "Query",
         "selections": [
           {
             "field": "ACCOUNTREFERENCE",
             "param": "accountId",
             "operand": "EQ",
             "required": "Y",
             "type": "string"
           }
         ]
       }
     ]
   }
 ],
 "version": "v1.0.0",
 "title": "account service",
 "key": "account-overview",
```

Service Definition

Title	account service	Key	account-overview
Version	v1.0.0	Schemes	[✓ https] http
Base Path	/api	Host	api.server.com

Service Endpoints

Path	/accounts/{accountId}
HTTP Method	GET
Operation	getAccountDetails
Operation Security	Public
Target	PZ.API.ACCOUNTS.1.0.0
Target Type	Query

Note: The inventory file can be copied directly from the Inventory pane and pasted into an IDE or if preferred click Finish to create a zip file containing the swagger spec, Camel routes, mappings and mock responses.

Click here to know how to create a zip file containing the swagger spec, Camel routes, mappings and mock responses.

Create Service Project

Create a service project using the irf-service-archetype.

Using the command line

mvn archetype:generate -DarchetypeGroupId=com.temenos.irf -DarchetypeArtifactId=irf-service-archetype

-DarchetypeVersion=1.0 -DarchetypeCatalog=local

Supply values for various arguments, either interactively or as parameters from the command line:

Define value for property 'groupId': com.temenos.training.irf

Define value for property 'artifactId': training-api

Define value for property 'version' 1.0-SNAPSHOT: :

Define value for property 'package' com.temenos.training.irf::

Confirm properties configuration:

groupId: com.temenos.training.irf

artifactId: training-api

version: 1.0-SNAPSHOT

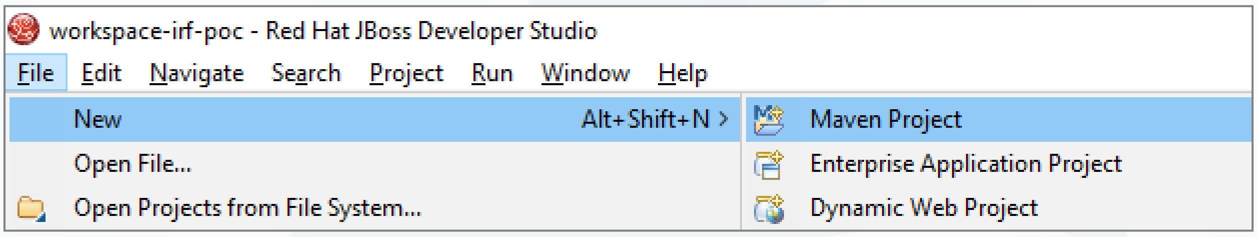
package: com.temenos.training.irf

As a single command:

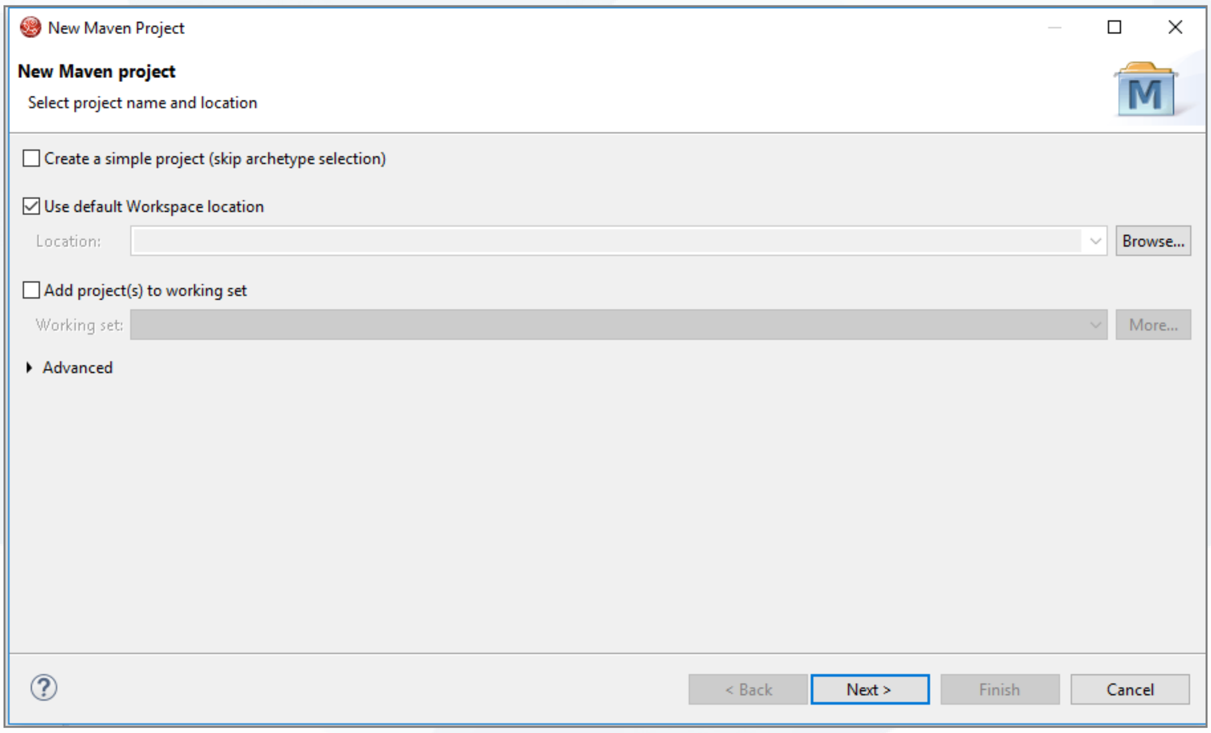
```
mvn archetype:generate -DarchetypeGroupId=com.temenos.irf -DarchetypeArtifactId=irf-service-archetype
-DarchetypeVersion=1.0 -DartifactId=training-api
-DgroupId=com.temenos.irf.training -Dpackage=com.temenos.irf.training -Dversion=SNAPSHOT-0.0.1
```

Using Eclipse

From the File menu, select **File** > **New** > **Maven Project**

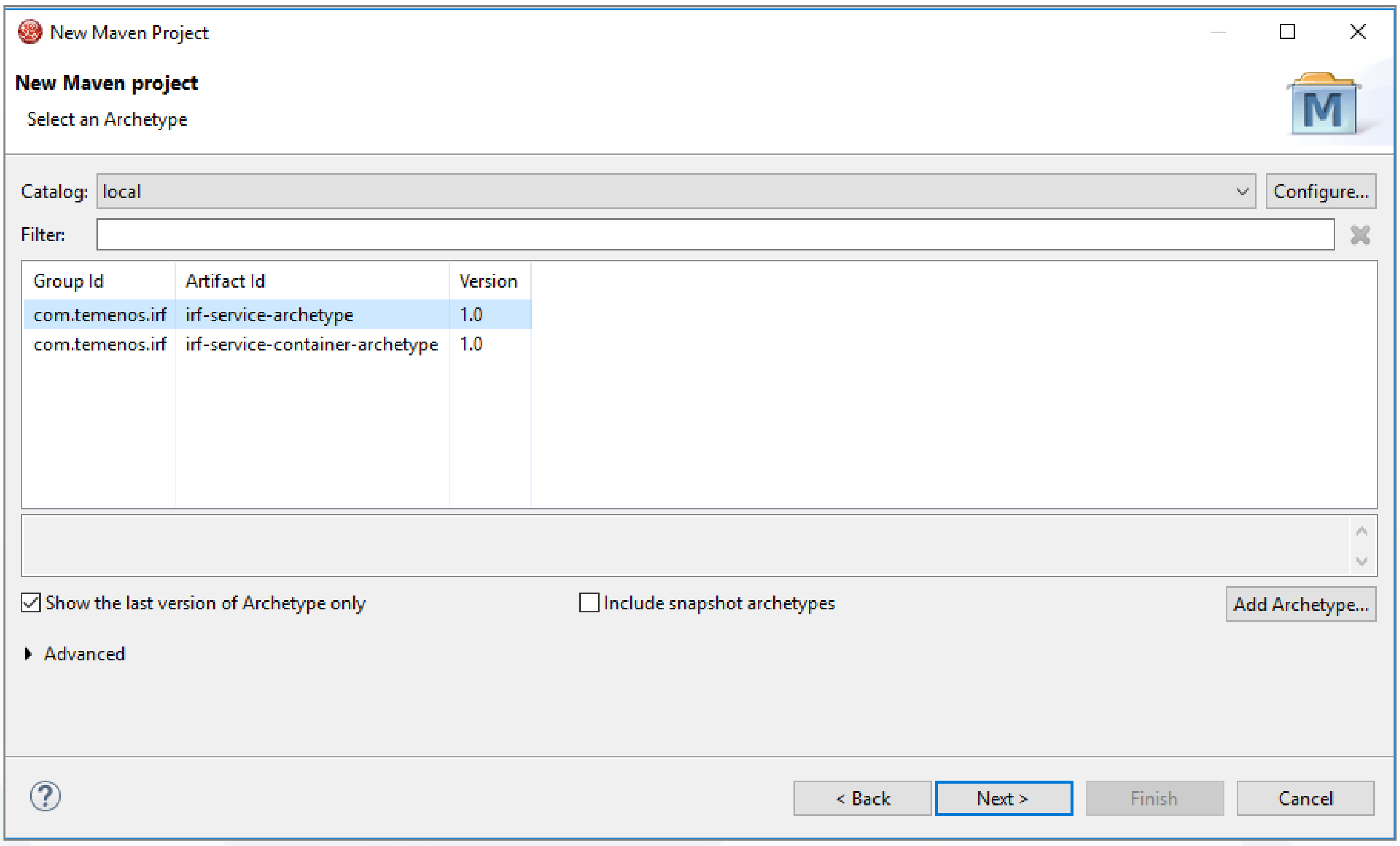

workspace-irf-poc - Red Hat JBoss Developer Studio
File
Edit
Navigate
Search
Project
Run
Window
Help
New
Alt+Shift+N
Open File...
Open Projects from File System...
Maven Project
Enterprise Application Project
Dynamic Web Project

Click the **Next** button.


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

Select archetype

Select the **irf-service-archetype** option from Artifact Id column and click the **Next** button.



Catalog: local

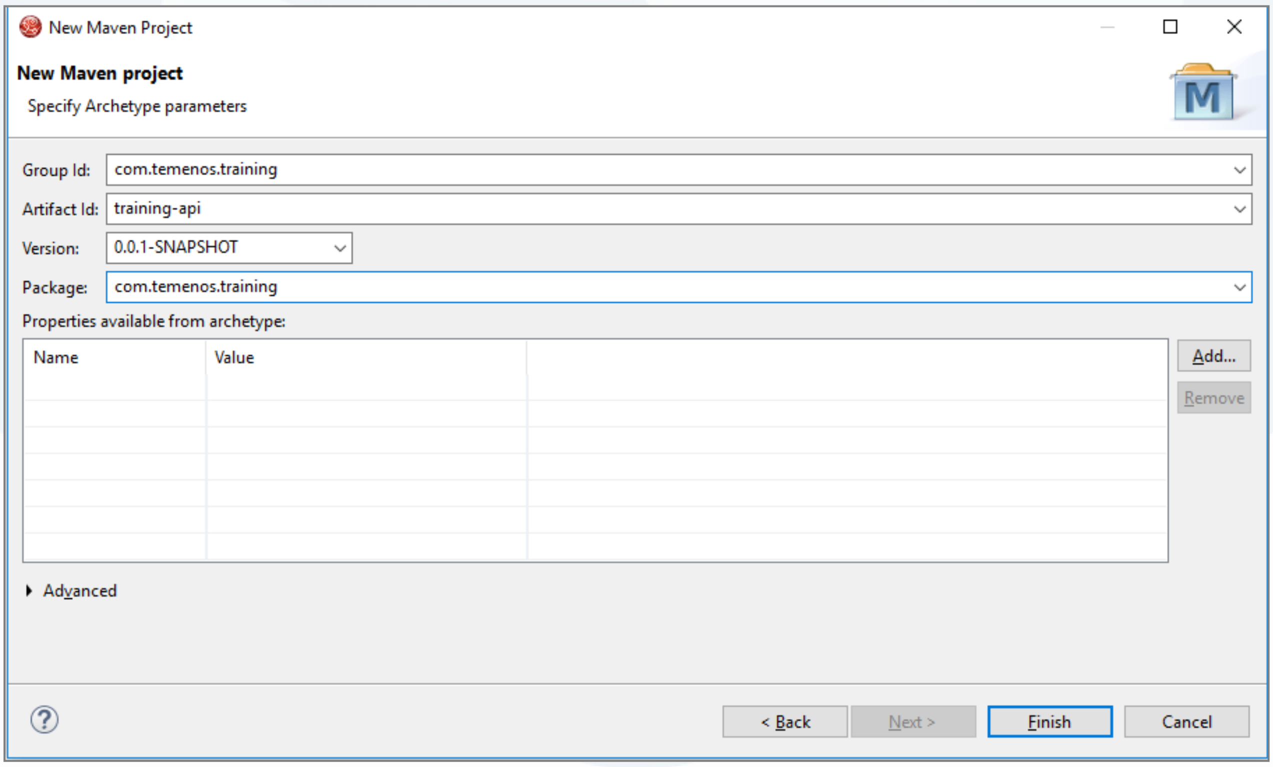
Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	1.0
com.temenos.irf	irf-service-container-archetype	1.0

☑ Show the last version of Archetype only
☐ Include snapshot archetypes

Provide Project details

Fill in the **Group Id**, **Artifact Id** and **Version** fields, and click the **Finish** button.



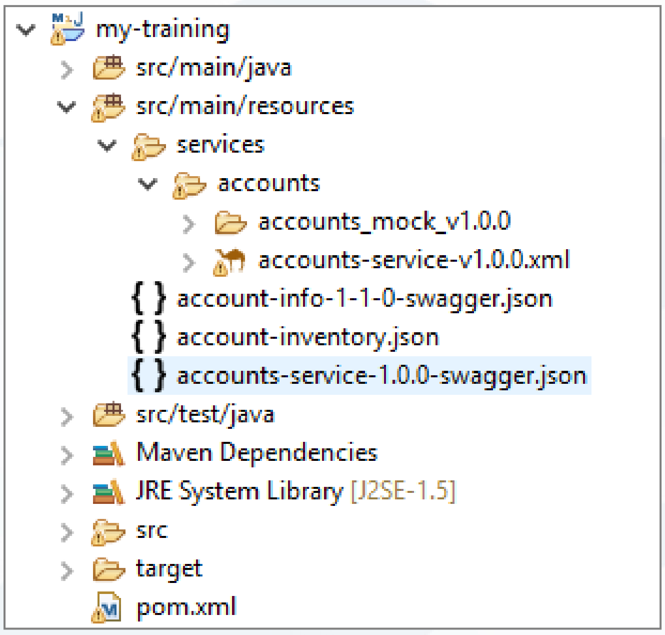
Group Id:	com.temenos.training
Artifact Id:	training-api
Version:	0.0.1-SNAPSHOT
Package:	com.temenos.training

Properties available from archetype:

Name	Value

Project Structure

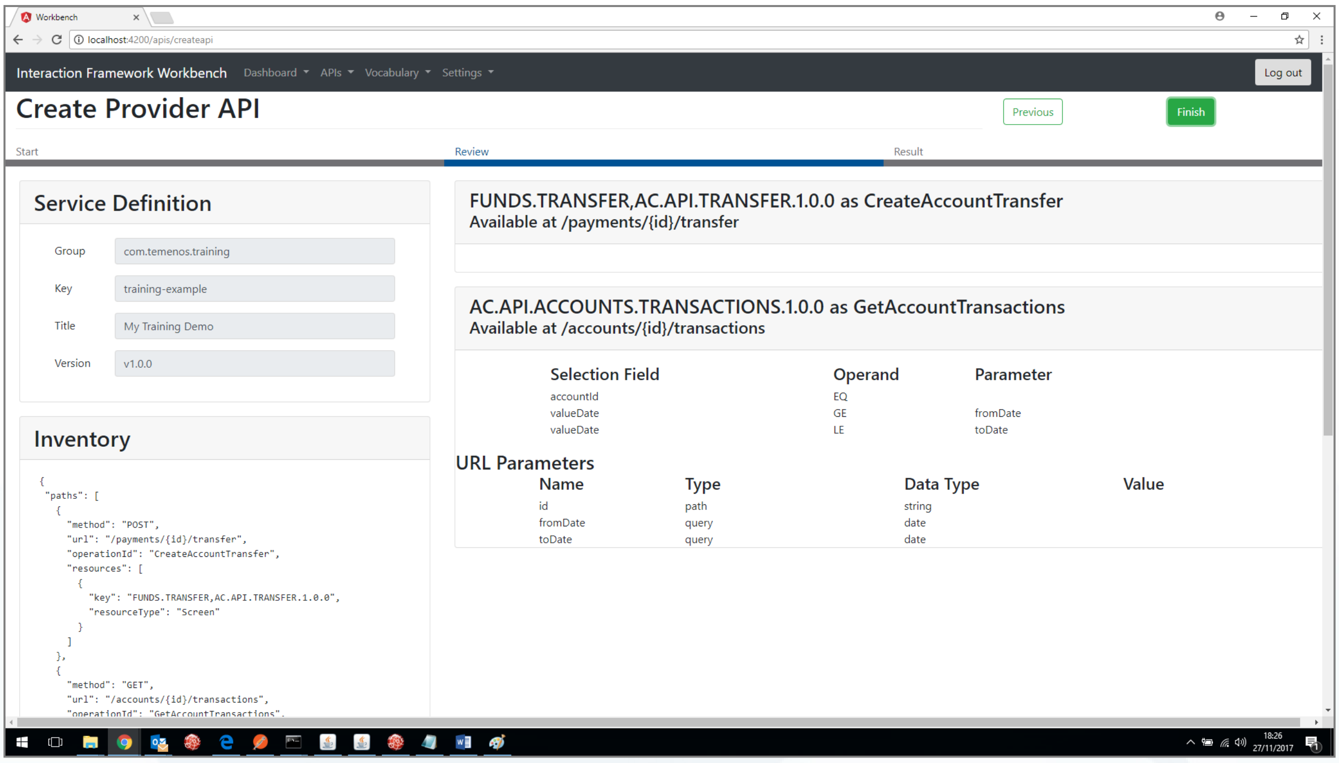
You should see a project created with the following structure:



```
my-training
  src/main/java
  src/main/resources
    services
      accounts
        accounts_mock_v1.0.0
        accounts-service-v1.0.0.xml
    account-info-1-1-0-swagger.json
    account-inventory.json
    accounts-service-1.0.0-swagger.json
  src/test/java
  Maven Dependencies
  JRE System Library [J2SE-1.5]
  src
  target
  pom.xml
```

Submit the Provider API

After creation of inventory file as shown in Step 4 of the Creating an Inventory using Interaction Framework Workbench.



Create Provider API

Service Definition

Group	com.temenos.training
Key	training-example
Title	My Training Demo
Version	v1.0.0

Inventory

```
{
 "paths": [
   {
     "method": "POST",
     "url": "/payments/{id}/transfer",
     "operationId": "CreateAccountTransfer",
     "resources": [
       {
         "key": "FUNDS.TRANSFER,AC.API.TRANSFER.1.0.0",
         "resourceType": "Screen"
       }
     ]
   },
   {
     "method": "GET",
     "url": "/accounts/{id}/transactions",
     "operationId": "GetAccountTransactions",
```

FUNDS.TRANSFER,AC.API.TRANSFER.1.0.0 as CreateAccountTransfer
Available at /payments/{id}/transfer

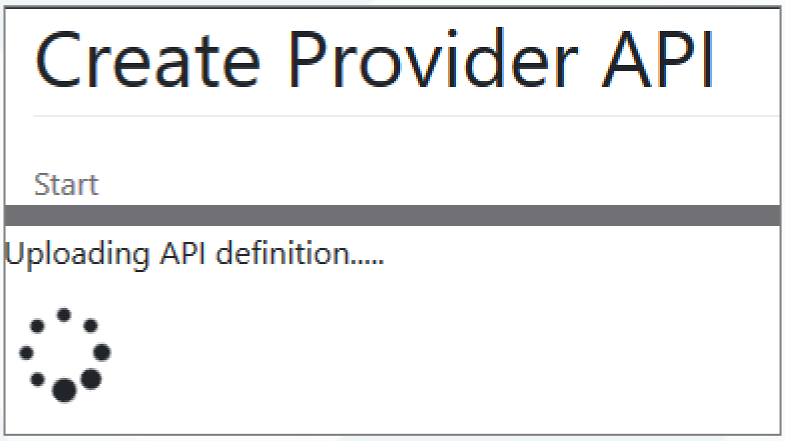
AC.API.ACCOUNTS.TRANSACTIONS.1.0.0 as GetAccountTransactions
Available at /accounts/{id}/transactions

Selection Field	Operand	Parameter
accountId	EQ	
valueDate	GE	fromDate
valueDate	LE	toDate

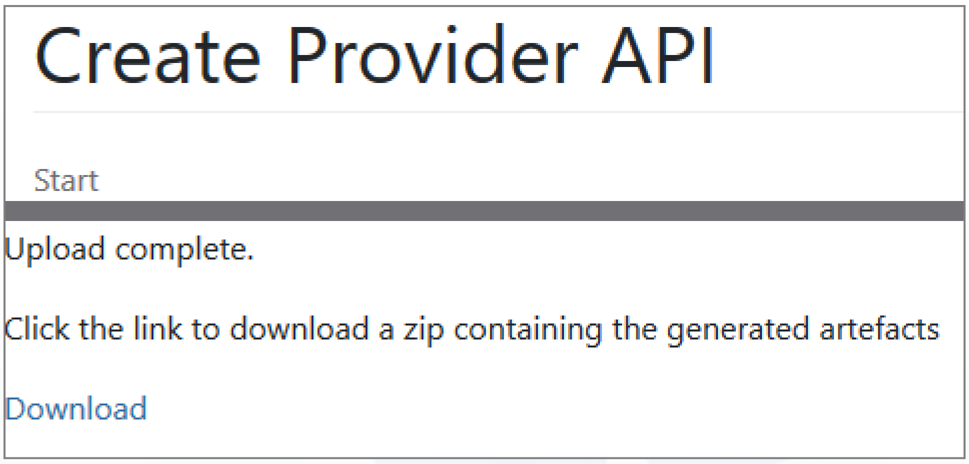
URL Parameters

Name	Type	Data Type	Value
id	path	string	
fromDate	query	date	
toDate	query	date	

Click Finish in the top right of the screen to submit the provider API.



The result provides a link to download a zip file containing the swagger spec, Camel routes, mappings and mock responses.


Create Provider API

Start

Upload complete.

Click the link to download a zip containing the generated artefacts

Download

Build the Service Project

Run the project as Maven > Install to build the project

Click here to know about **T24 Connectivity Configuration TAFJ and TAFC**

Deploy and Test Provider API

Add to the Service Container Project

In your service container project, amend the pom.xml to include your API as a dependency:

NB There are several dependencies tags in the pom.xml file - this dependency needs to be added to the MAIN dependency, below the comment:

<!-- Add any project dependencies here-->

```
<dependency>

<groupId>com.temenos.training</groupId>

    <artifactId>provider-accounts</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>
```

The service container should restart automatically, if not stop the server and restart manually.

Check Deployment

Use Postman to make a GET request to the management service

```
localhost:8080/api/v1.0.0/meta/apis
```

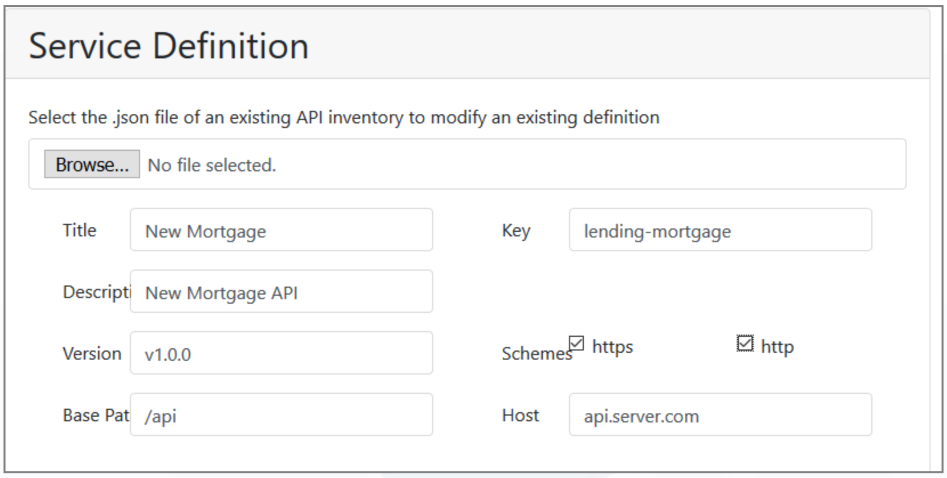
You should see the servicer returned in the response:

```
{

    "header": {
        "serviceCount": 9,

        "audit": {

            "processTime": 9

        }

    },

    "body": [

        {

            "service": "provider-accounts.service.v1.0.0",

            "endPoints": [

                {

                   "method": "POST",

                    "uri": "/v1.0.0/provider-accounts/payments"

                },

                {

                    "method": "GET",

                    "uri": "/v1.0.0/provider-accounts/{accountId}/transactions"

               }

            ]

        },

        {

            "service": "accounts.service.v1.0.0",

            "endPoints": [

                {

                    "method": "GET",

                    "uri": "/v1.0.0/accounts/{AccountId}/transactions"

               }

            ]

        },
```

Exposing AA Activities as Provider APIs

Service configuration

Define the Service Definition parameters for "New Arrangement creation" for a Mortgage loan product.



Service Definition

Select the .json file of an existing API inventory to modify an existing definition

Browse... No file selected.

Title	New Mortgage	Key	lending-mortgage
Descripti	New Mortgage API		
Version	v1.0.0	Schemes	☑ https ☑ http
Base Pat	/api	Host	api.server.com

Select the "LENDING" option under the "Available artifact" section and add it.

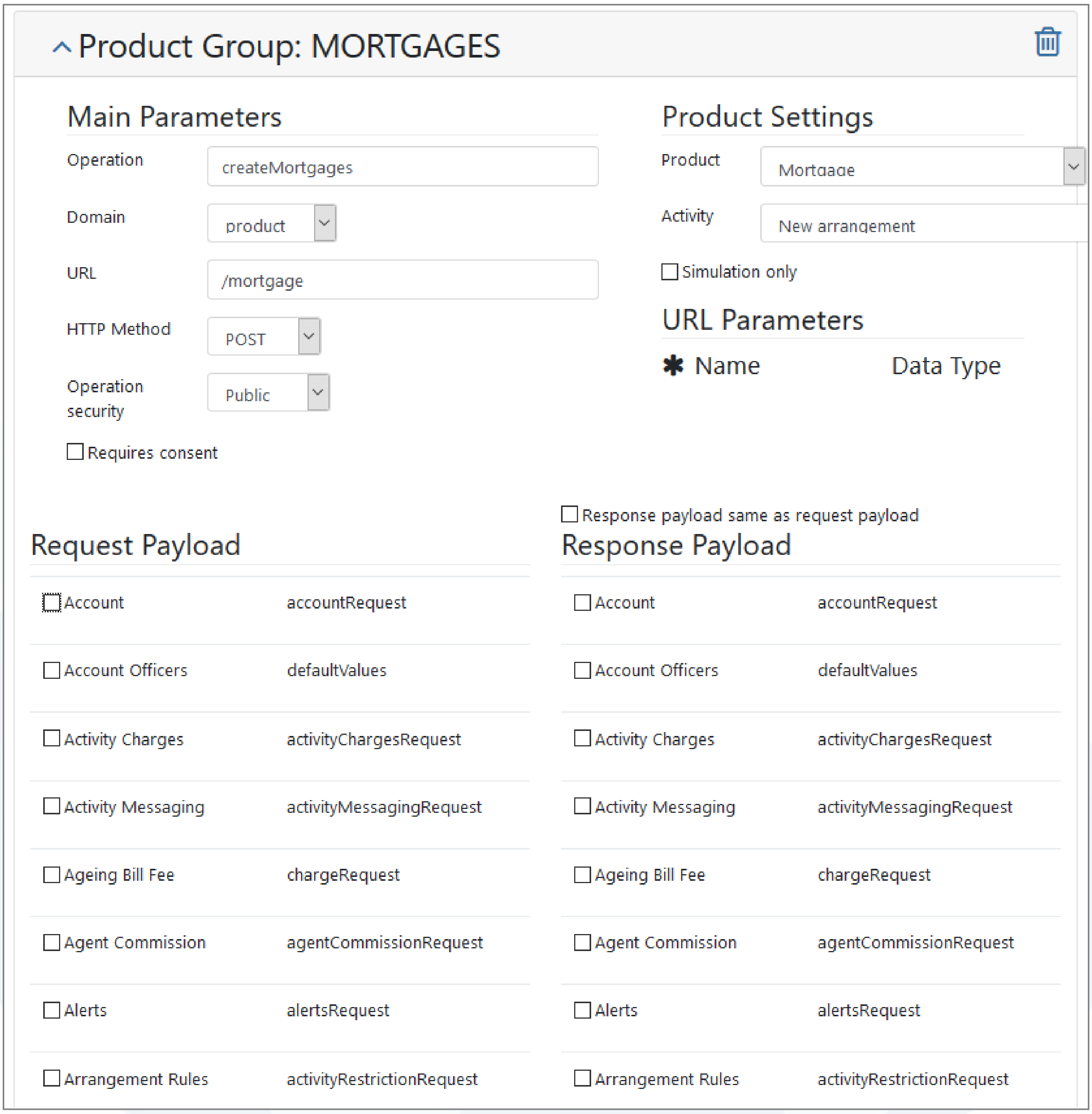
Enter Activity & Product Details

Select the product group **MORTGAGES** from the list and click Product: **Mortgage** and Activity: **New Arrangement**.

Select-able payload properties should appear on the screen.

Click the request and response payload tick boxes to specify the input (**Request Payload**) and output (**Response Payload**). In case both of them are the same, click "**Response payload same as request payload**".

Default behavior, i.e., in case no boxes are selected, will result in all the applicable properties to be used. Response payload version can be configured by creating **AA.ARR.{property class},AA.API.RESPONSE.1.0.0** For example: **AA.ARR.ACCOUNT,AA.API.RESPONSE.1.0.0**


Product Group: MORTGAGES

Main Parameters

Operation	createMortgages
Domain	product
URL	/mortgage
HTTP Method	POST
Operation security	Public

☐ Requires consent

Product Settings

Product	Mortgage
Activity	New arrangement

☐ Simulation only

URL Parameters

✱ Name	Data Type

☐ Response payload same as request payload

Request Payload

☐ Account	accountRequest
☐ Account Officers	defaultValues
☐ Activity Charges	activityChargesRequest
☐ Activity Messaging	activityMessagingRequest
☐ Ageing Bill Fee	chargeRequest
☐ Agent Commission	agentCommissionRequest
☐ Alerts	alertsRequest
☐ Arrangement Rules	activityRestrictionRequest

Response Payload

☐ Account	accountRequest
☐ Account Officers	defaultValues
☐ Activity Charges	activityChargesRequest
☐ Activity Messaging	activityMessagingRequest
☐ Ageing Bill Fee	chargeRequest
☐ Agent Commission	agentCommissionRequest
☐ Alerts	alertsRequest
☐ Arrangement Rules	activityRestrictionRequest

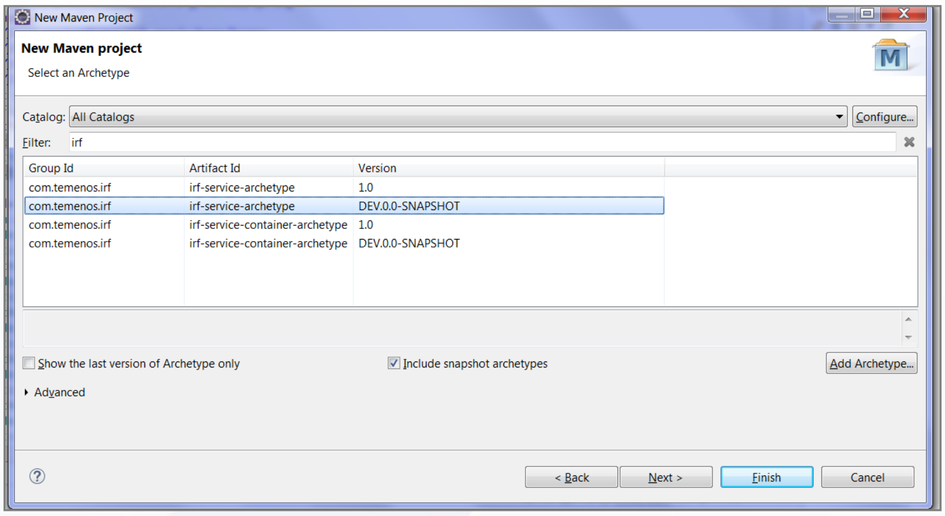
Download the Generated Artifacts

Click Next and click Finish.

Workbench displays a URL link to download the generated artifacts.

Create Lending API project using eclipse Maven Archetype

1. Create provider api project for Lending Product using the Maven Arch type.
   a. Select "**irf-service-archetype**" of Dev.0.0.0-Snapshot Version



New Maven project

Select an Archetype

Catalog: All Catalogs

Filter: irf

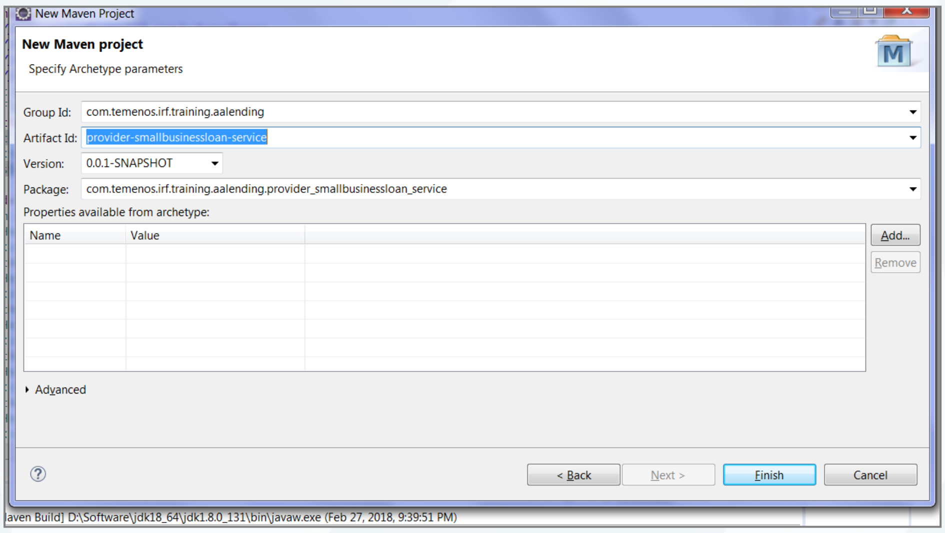
Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	1.0
com.temenos.irf	irf-service-archetype	DEV.0.0-SNAPSHOT
com.temenos.irf	irf-service-container-archetype	1.0
com.temenos.irf	irf-service-container-archetype	DEV.0.0-SNAPSHOT

☐ Show the last version of Archetype only ☑ Include snapshot archetypes

   b. Enter Service API project details

      Group Id : com.temenos.irf.training.aalending

      Artifact id : provider-smallbusinessloan-service



New Maven project

Specify Archetype parameters

Group Id: com.temenos.irf.training.aalending

Artifact Id: provider-smallbusinessloan-service

Version: 0.0.1-SNAPSHOT

Package: com.temenos.irf.training.aalending.provider_smallbusinessloan_service

Properties available from archetype:

Name	Value

aven Build] D:\Software\jdk18_64\jdk1.8.0_131\bin\javaw.exe (Feb 27, 2018, 9:39:51 PM)

Click Finish.

Import and Build the Generated Artifacts

1. Extract the generated artifacts into the resources/services folder of the Lending API project created from the Service API maven archetype.
2. Buildthe Service API project using maven command
   mvn clean install

3. Add dependencies of API maven project in the Container maven pom.xml under "dependencies" section.

```
<dependency>
<groupId>com.temenos.irf.training.aalending</groupId>
<artifactId>provider-smallbusinessloan-service</artifactId>
<version>0.0.1-SNAPSHOT</version>
</dependency>
```

4. Build and Restart the Container project

Test Mortgage Loan Service

1. Test the Service from Postman using the below Url
   a. URL :http://localhost:8080/api/v1.0.0/product/mortgage
   b. data :

```
{

"header": {



},

"body": {

"parties": [

{

"partyId": "100100",

"partyRole": "USD"

}

],

"currencyId": "USD",

"commitment": {

"amount": "7000",

"term": "1Y"

}

}

}
```

   and Post the request

2. Get success response with the Arrangement details

Replace Indicator

By default, all PUT request messages are mapped with OFS replace Indicator.

Messages (update messages) with replace indicator tells T24 to update the record by regenerating the entire record. Hence, the mandatory fields required by the record generation process has to be passed to T24 for a successful output. To learn more about OFS replace indicator, please refer to OFS User Guides.

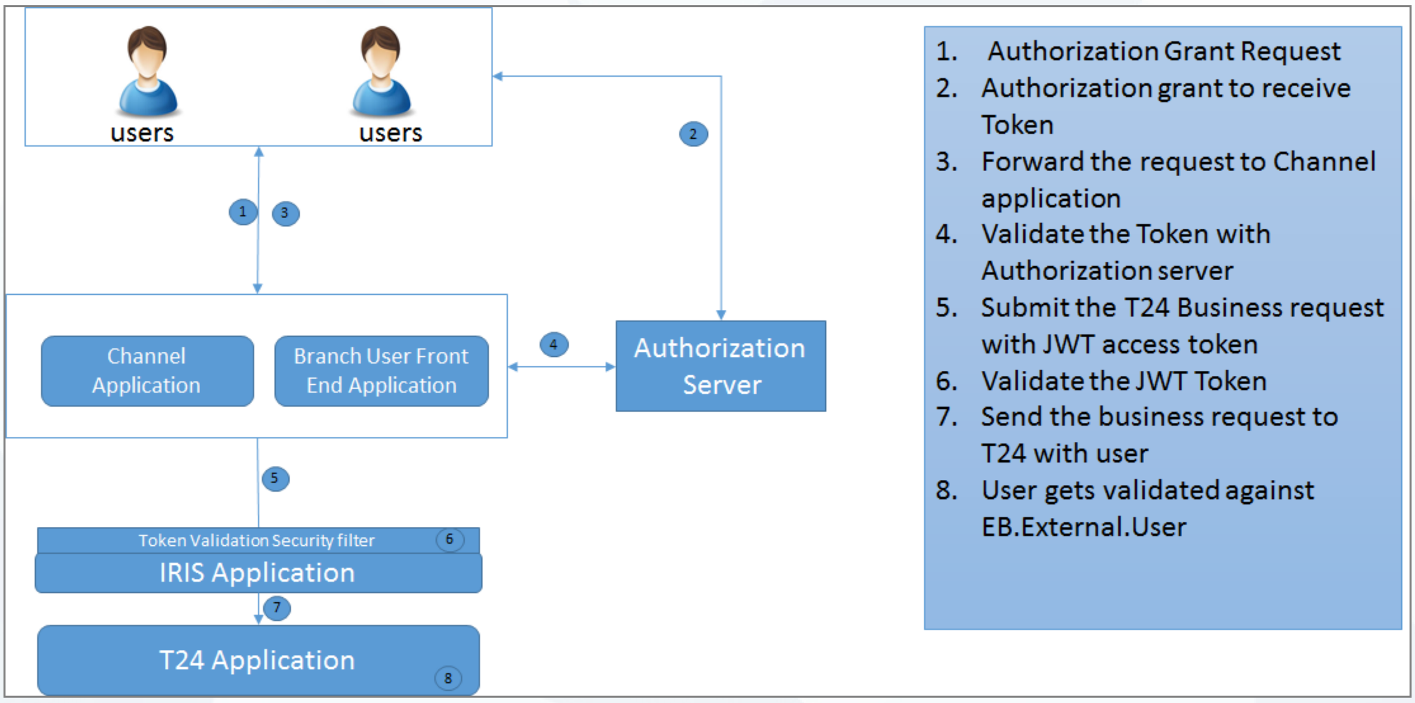
To disable replace indicator functionality, add the following in your service XML.

```
...

<setProperty propertyName="ignoreReplace">

<constant>true</constant>

</setProperty>

...
```

Security

Token Based Security Filter Implementation

IRIS Data Services can be supported as an intermediary in an authenticated request with the JWT token as a mechanism for authentication and identification with the help of this functionality. The handling of the JWT token is expected to be fully configurable to support various deployment choices from naive authenticator to the most secure one.



1. Authorization Grant Request
2. Authorization grant to receive Token
3. Forward the request to Channel application
4. Validate the Token with Authorization server
5. Submit the T24 Business request with JWT access token
6. Validate the JWT Token
7. Send the business request to T24 with user
8. User gets validated against EB.External.User

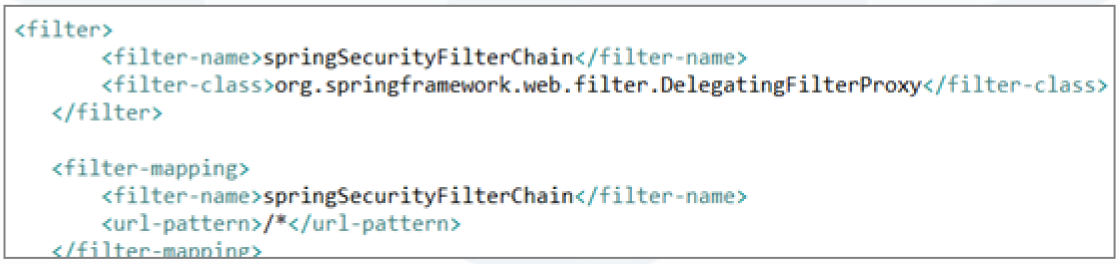
The JWT Token based authentication solution is designed as pluggable and extendable through configuration.

Click here to go to the **Configurations** page.

Configurations

Spring Security Filter Enablement

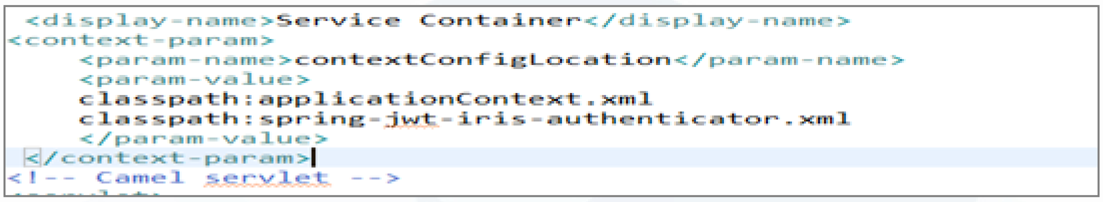
IRIS web application gets enabled with Spring security filter chains by configuring respective tags in the web.xml of IRIS application. **springSecurityFilterChain** Filter should be added to enable spring security.



```
<filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Spring Security Configurations

spring-jwt-iris-authenticator.xml – is used to define spring beans and should be loaded as a part of configuration. Add spring configuration xml in the web configuration settings.

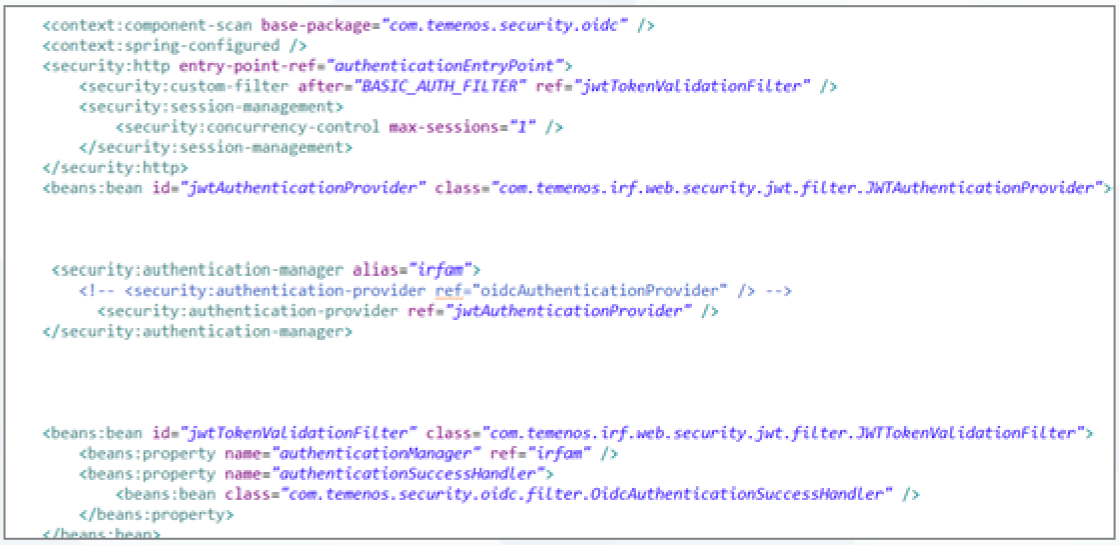


```
 <display-name>Service Container</display-name>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
    classpath:applicationContext.xml
    classpath:spring-jwt-iris-authenticator.xml
    </param-value>
 </context-param>
<!-- Camel servlet -->
```

spring-jwt-iris-authenticator.xml Configuration

This authenticator xml allows the user to extend the IRIS infrastructure based on their token validation requirements.

Below section helps to configure the Spring Security Authentication Provider and the Token Validation filter based on the user requirement.



```
<context:component-scan base-package="com.temenos.security.oidc" />
<context:spring-configured />
<security:http entry-point-ref="authenticationEntryPoint">
    <security:custom-filter after="BASIC_AUTH_FILTER" ref="jwtTokenValidationFilter" />
    <security:session-management>
        <security:concurrency-control max-sessions="1" />
    </security:session-management>
</security:http>
<beans:bean id="jwtAuthenticationProvider" class="com.temenos.irf.web.security.jwt.filter.JWTAuthenticationProvider">


 <security:authentication-manager alias="irfam">
    <!-- <security:authentication-provider ref="oidcAuthenticationProvider" /> -->
      <security:authentication-provider ref="jwtAuthenticationProvider" />
</security:authentication-manager>



<beans:bean id="jwtTokenValidationFilter" class="com.temenos.irf.web.security.jwt.filter.JWTTokenValidationFilter">
    <beans:property name="authenticationManager" ref="irfam" />
    <beans:property name="authenticationSuccessHandler">
        <beans:bean class="com.temenos.security.oidc.filter.OidcAuthenticationSuccessHandler" />
    </beans:property>
</beans:bean>
```

Out of the box, Temenos provides Validation filter and Authentication provider to validate the JWT token. The Validation filter uses this configuration XML to perform the algorithm, claim, and Signature validations.

I. **AlgorithmValidator**

   The ‘alg’ value should be the default of RS256 or any one of the values given under RelyingPartyClient property “idTokenSignedAlg”.

   The ‘kid’ optional parameter, if present, is used to identify the key for decrypting the token applying the valid algorithm in the ‘alg’ claim.

II. **SignatureValidator**

   A signed token needs to be validated for the signature to verify the integrity of the token.

If the client is configured to support an unsigned token and there is no signature then there is no signature validation required.

III. **ClaimValidator**

ClaimValidator is a generic validator for various claims in the token. Each instance of the ClaimValidator would validate one specific claim. The claim that needs to be validated is injected into the instance.

The following are the claims default validators supported for:

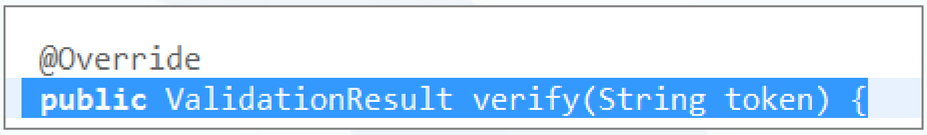
1. exp
2. iss
3. aud
4. azp
5. iat

IV. **CustomValidator**

A generic custom validator can be defined through which any number of custom validations could be written and plugged into the validator composite through simple spring beans configuration.

For example, if the user wants to implement a custom validator to perform custom validation with authentication provider, user can implement additional validator by implementing base interface of the validation which provides token and the http request objects to the custom validator.

Custom validator needs to extend "**com.temenos.security.oidc.token.validator. TokenValidator**" and should implement "**verify**" method



```
@Override
public ValidationResult verify(String token) {
```

Upon successful authentication by obtaining and validating the token, the validation filter builds and propagates the authenticated principal and invokes the remaining filter change as normal.

Spring Security uses **org.springframework.security.core.userdetails.UserDetails** to manage user information across the application. Once the requests get authenticated, Spring security made userDetails available as a part of Spring Authentication object.

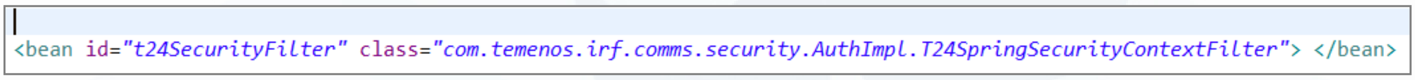
IRISAuthenticatedUser is the iris authenticated user object used to carry user data as a part of Spring Authentication object.

On successful authentication of the token, the successfulAuthentication callback sets Spring Authentication object references in the Spring Security Context. The Security Context holds the user information (IRISAuthenticatedUser) which is available across the IRIS Version and Enquiry processors.

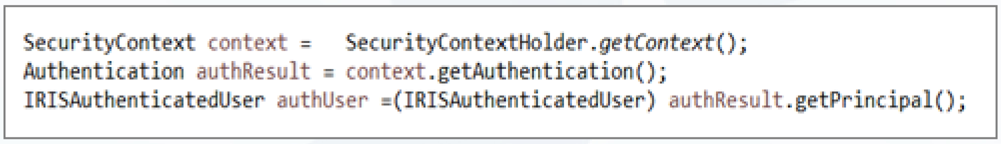
T24 Security Filter Configuration

IRIS T24 Processors uses T24 Security Context to construct OFS message with the respective user credentials. T24SecurityFilter derives the User principal from the Spring Security Context and constructs user's T24 security context.

Out of the box, Temenos provides a T24 Security filter "com.temenos.irf.comms.security.AuthImpl.T24SpringSecurityContextFilter". This T24 Security filter retrieves the user principal from spring security context and constructs T24 Security context which is used for constructing the OFS message.



```
<bean id="t24SecurityFilter" class="com.temenos.irf.comms.security.AuthImpl.T24SpringSecurityContextFilter"> </bean>
```



```
SecurityContext context =   SecurityContextHolder.getContext();
Authentication authResult = context.getAuthentication();
IRISAuthenticatedUser authUser =(IRISAuthenticatedUser) authResult.getPrincipal();
```

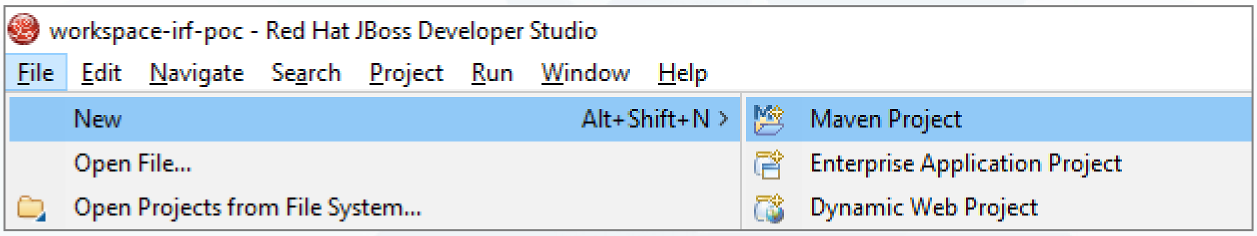
Creation of Published API

Overview

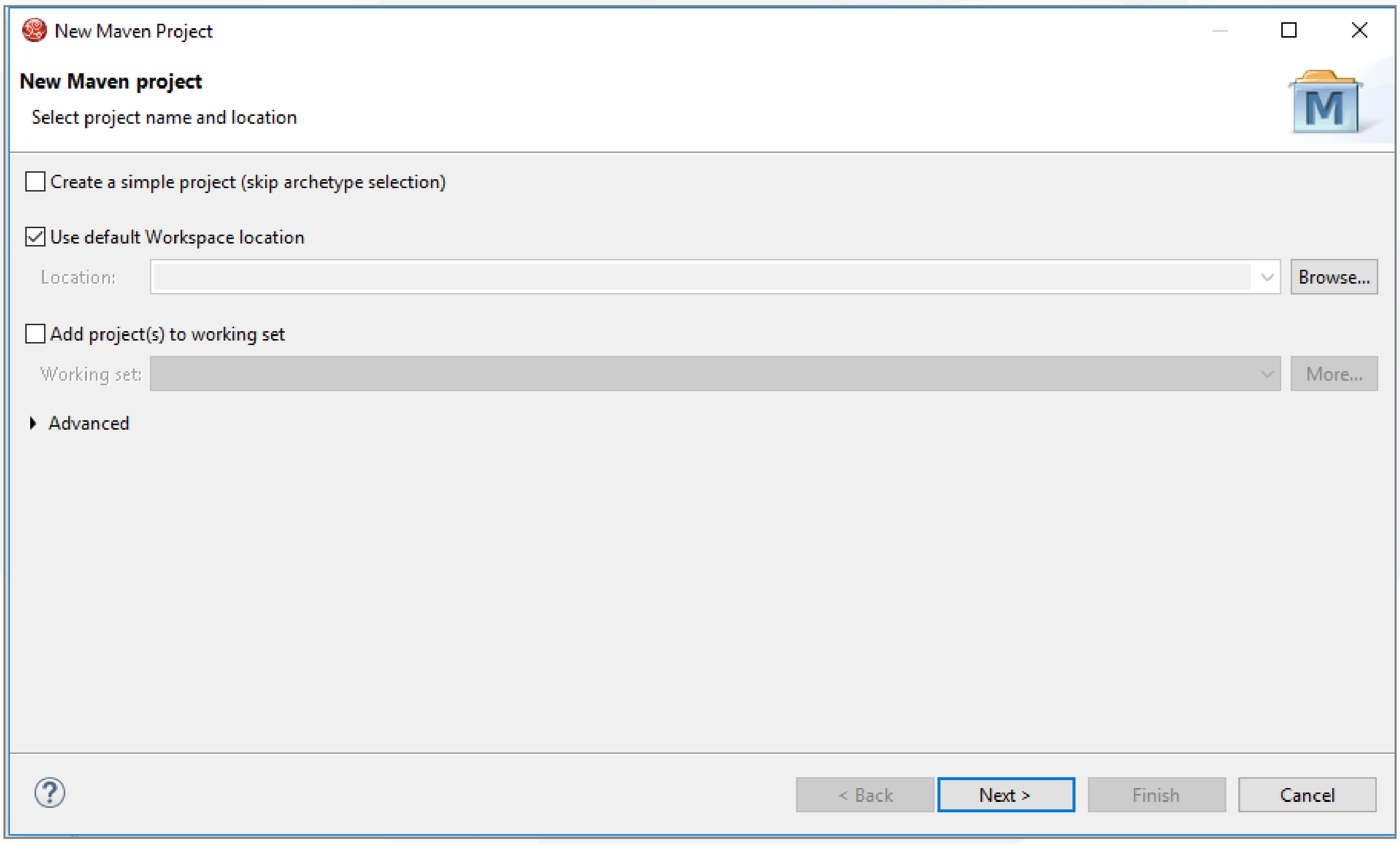
An API defined in Swagger 2.0 may be imported into an API project. The import process will create the Camel routes, mock responses and wire the routes to the mock responses. This is done using a Maven plugin, either from your IDE or from the command line.

Creation of Provider Service Project

From the File menu, select **File** > **New** > **Maven Project**

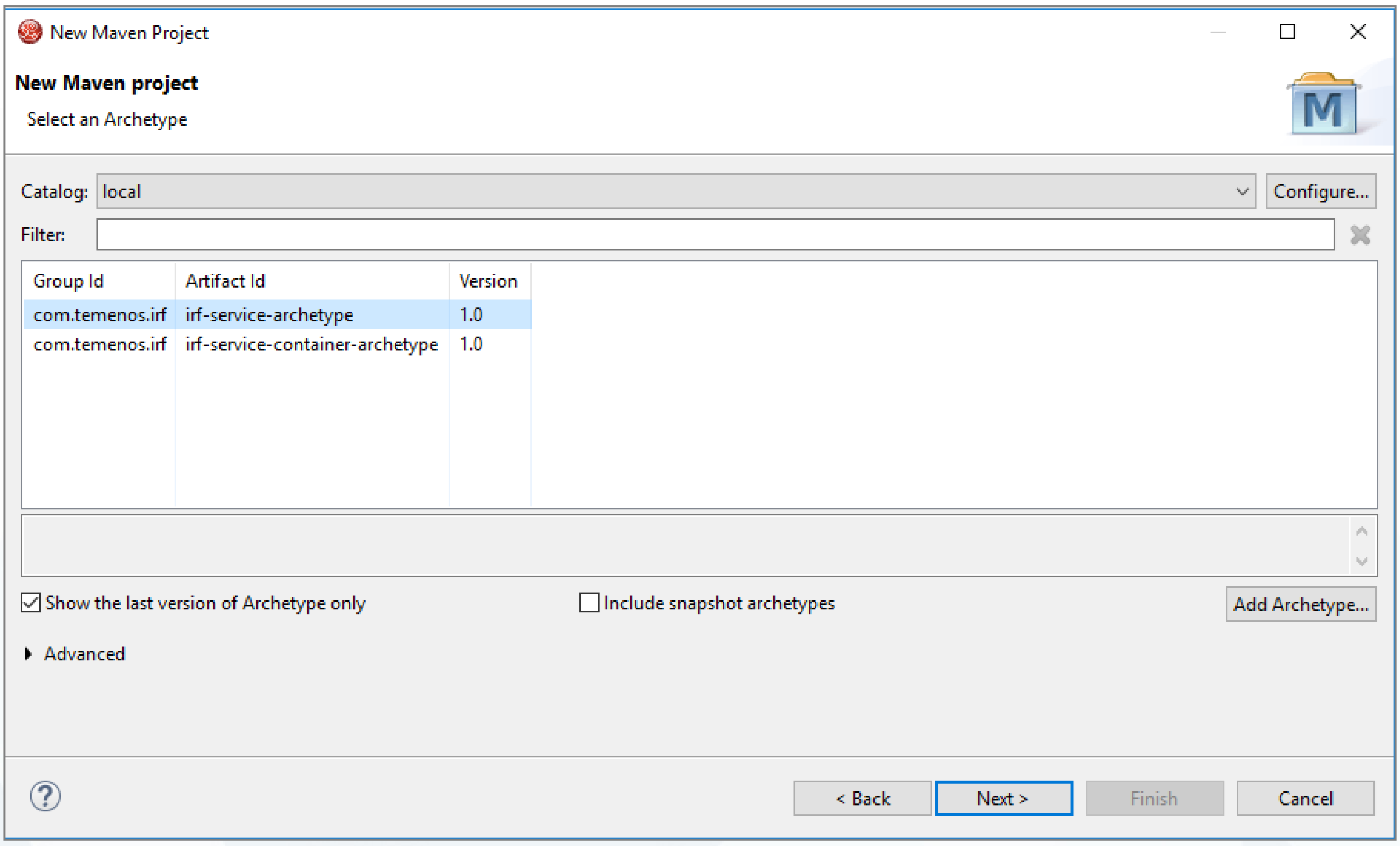

workspace-irf-poc - Red Hat JBoss Developer Studio
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Maven Project
Enterprise Application Project
Dynamic Web Project

Click **Next**


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

Select archetype

Select the **irf-service-archetype** and click **next**



Catalog: local

Filter:

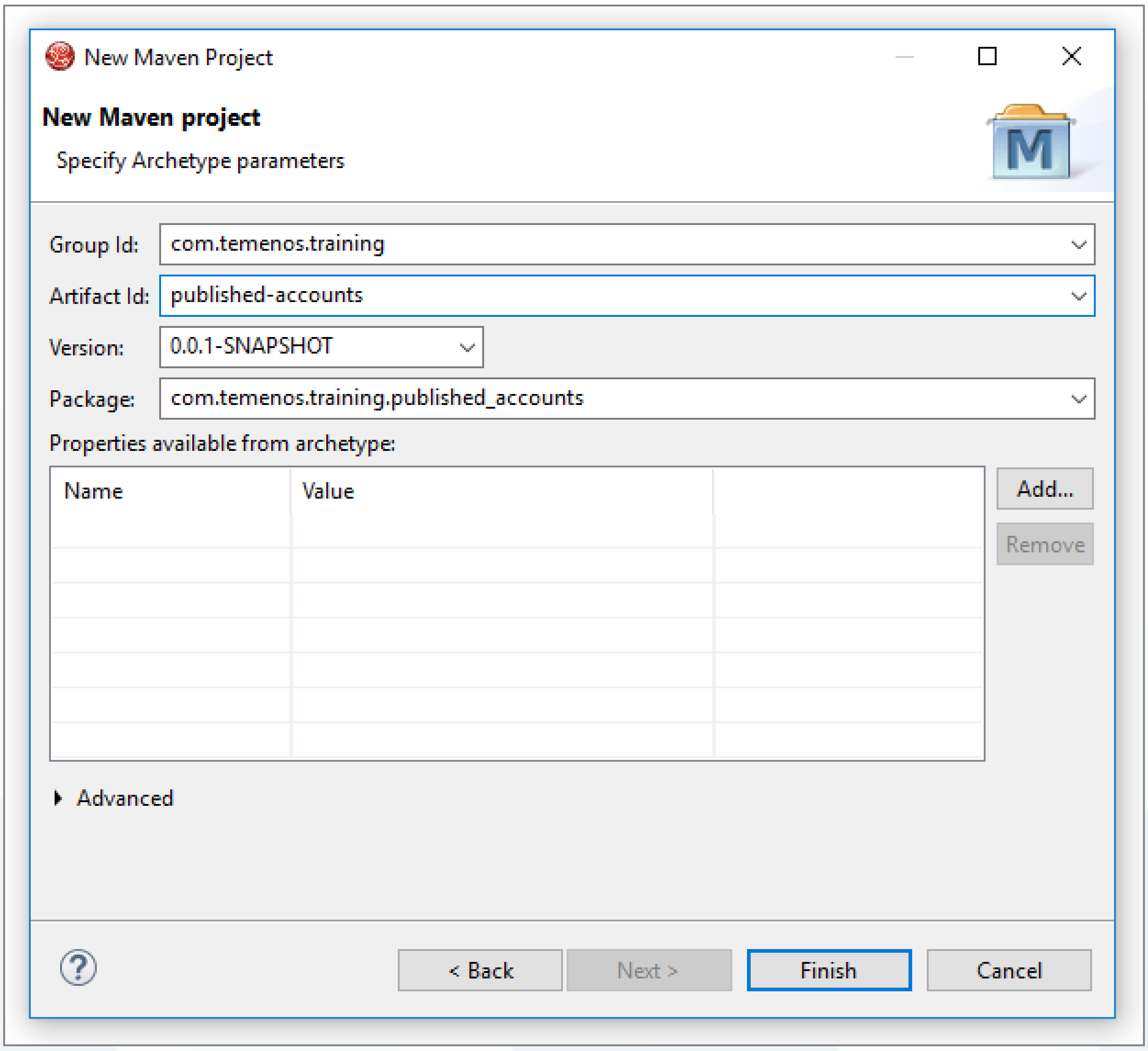
Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	1.0
com.temenos.irf	irf-service-container-archetype	1.0

☑ Show the last version of Archetype only

☐ Include snapshot archetypes

Provide project details

Fill in the **Group Id**, **Artifact Id** and **Version** fields, and click **Finish**.



New Maven Project

New Maven project

Specify Archetype parameters

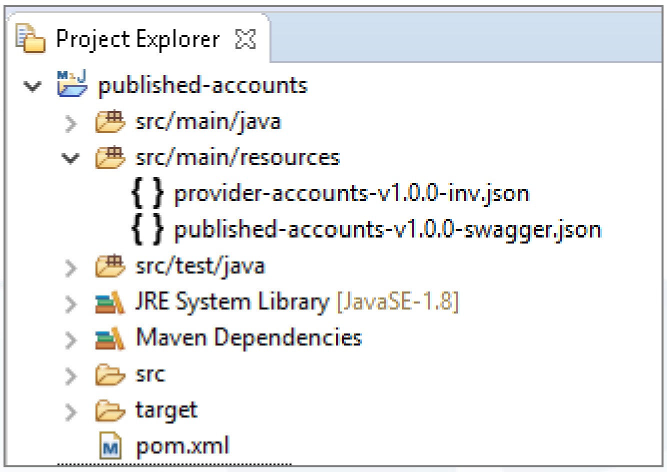
Group Id:	com.temenos.training
Artifact Id:	published-accounts
Version:	0.0.1-SNAPSHOT
Package:	com.temenos.training.published_accounts

Properties available from archetype:

Name	Value

Project Structure

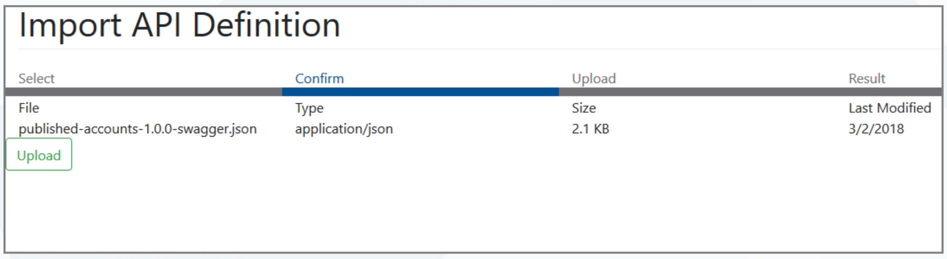
You should see a project created with the following structure:


Project Explorer

published-accounts
- src/main/java
- src/main/resources
  - provider-accounts-v1.0.0-inv.json
  - published-accounts-v1.0.0-swagger.json
- src/test/java
- JRE System Library [JavaSE-1.8]
- Maven Dependencies
- src
- target
- pom.xml

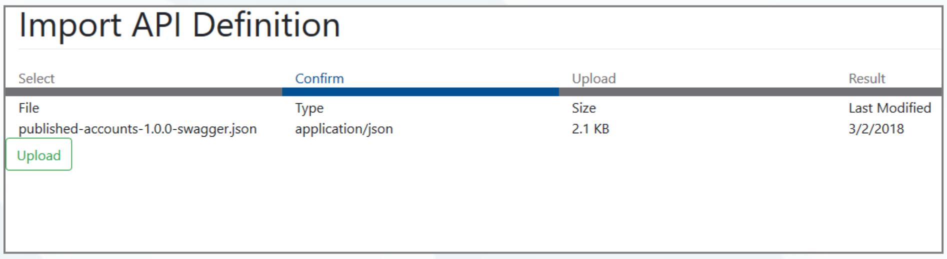
Import API Spec

The import process is done using the Interaction Framework Workbench. Go to APIs > Import API Definition. You should be able to see the following screen.


Import API Definition

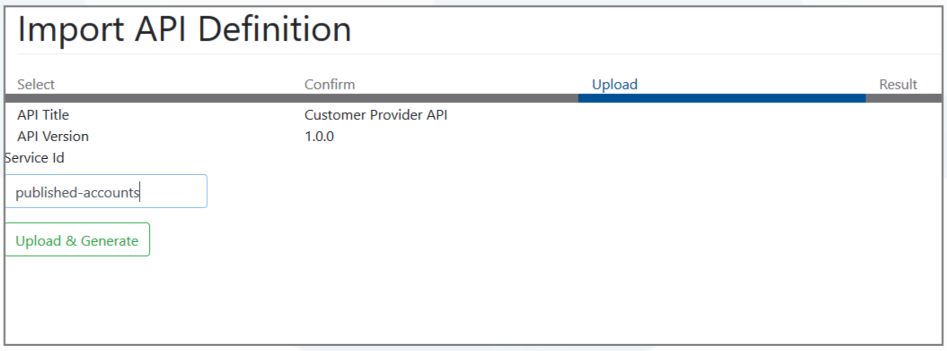
File	Type	Size	Last Modified
published-accounts-1.0.0-swagger.json	application/json	2.1 KB	3/2/2018

Select the publisher swagger definition (swagger json file) that you want to import and then click Upload


Import API Definition

File	Type	Size	Last Modified
published-accounts-1.0.0-swagger.json	application/json	2.1 KB	3/2/2018

Give your service a name in the next screen and click Upload & Generate


Import API Definition

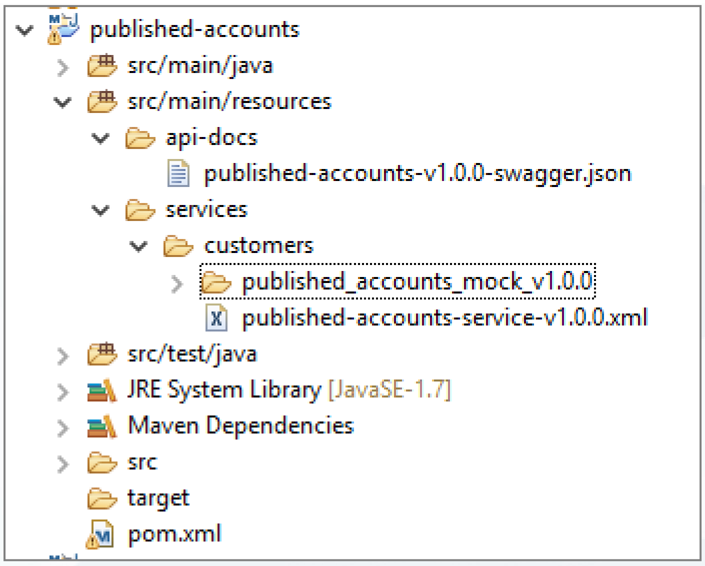
API Title	Customer Provider API
API Version	1.0.0

Service Id: published-accounts

Download and extract the zip and place the generated files in the service project

Project Structure

Refresh the project view and verify the project structure:



```
published-accounts
  src/main/java
  src/main/resources
    api-docs
      published-accounts-v1.0.0-swagger.json
    services
      customers
        published_accounts_mock_v1.0.0
        published-accounts-service-v1.0.0.xml
  src/test/java
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  src
  target
  pom.xml
```

Provider API to Published API Mapping

Map path params from publisher to provider

Route for published Funds Transfer Status Report

```
...

<rest path="/v1.0.0/published" produces="application/json" id="published.service.restlet">

<get uri="/payments/{accountid}/statusreport" id="PublishedPaymentStatusReport">

<param name="accountid" type="path" required="true"/>

<to uri="direct:published.PublishedPaymentStatusReport"/>

</get>

</rest>

...

<route id="direct.published.PublishedPaymentStatusReport">

<from uri="direct:published.PublishedPaymentStatusReport"/>

<to uri="direct-vm:order-PSD2.getPaymentTransferList" />

<process ref="FTStatusReportResponseMapper"/>

</route>
```

Note: Use "direct-vm" component to map all the publisher path params to provider path params. This will enable it to look through all the CamelContexts present in the same JVM and route to the underlying provider API.

Save all the files, and build the publisher api project as Maven > Install.

Sample Publisher route for the reference

Additional Beans before CamelContext tag

```
...

<bean id="FTInitiationRequestMapper" class="com.temenos.irf.published_FT_api.FTIni-
tiationRequestProcessor"/>

<bean id="FTInitiationResponseMapper" class="com.temenos.irf.published_FT_api.FTIni-
tiationResponseProcessor"/>

<bean id="FTStatusReportResponseMapper" class="com.temenos.irf.published_FT_api.FTStatusRe-
portResponseProcessor"/>

<bean id="FTSubmissionResponseMapper" class="com.temenos.irf.published_FT_api.FTSub-
missionResponseProcessor"/>

...
```

Published FT Status Report Route

```
...

<get uri="/payments/{accountid}/statusreport" id="PublishedPaymentStatusReport">

<param name="accountid" type="path" required="true"/>

<to uri="direct:published.PublishedPaymentStatusReport"/>

</get>

...

<route id="direct.published.PublishedPaymentStatusReport">

<from uri="direct:published.PublishedPaymentStatusReport"/>

<to uri="direct-vm:order-PSD2.getPaymentTransferList" />

<process ref="FTStatusReportResponseMapper"/>

</route>
```

Published FT Initiation Route

```
...

<post uri="/payments/transfer/initiation" id="PublishedPaymentTransferInitiation">
```

```
<to uri="direct:published.PublishedPaymentTransferInitiation"/>

</post>

...

<route id="direct.published.PublishedPaymentTransferInitiation">

<from uri="direct:published.PublishedPaymentTransferInitiation"/>

<process ref="FTInitiationRequestMapper"/>

<to uri="direct-vm:order-PSD2.getPaymentTransferList" />

<process ref="FTInitiationResponseMapper"/>

</route>
```

Published FT Submission Route

```
...

<post uri="/payments/transfer/{id}/submission" id="PublishedPaymentTransferSubmission">

<param name="id" type="path" required="true"/>

<to uri="direct:published.PublishedPaymentTransferSubmission"/>

</post>

...

<route id="direct.published.PublishedPaymentTransferSubmission">

<from uri="direct:published.PublishedPaymentTransferSubmission"/>

<to uri="direct-vm:order-PSD2.getPaymentTransferList" />

<process ref="FTSubmissionResponseMapper"/>

</route>
```

The request and response mappers configured will have java code to transform from publisher json format to provider json and vice versa.

Deployment and Configuration

This section covers the following topics.

Deployment options

Once the service project is build, the user can deploy it using either of the following application servers:

- Jboss
- Weblogic
- Websphere

You can configure the IRIS R18 solution using TAFJ (JMS Connector) or TAFC (TCS and TOCFEE Connectors).

T24 Connectivity Configuration TAFJ and TAFC

The following connectors are used in connectivity configuration of TAFJ and TAFC:

JMS Connector - TAFJ & TAFC TOCFEE

From T24 R09, the Temenos Open Connectivity Framework has been updated to allow deployment in Java Enterprise Edition application servers – TOCF (EE). TOCF (EE) is a family of JEE compliant components providing connectivity to T24. However, in addition to utilising industry standard connectivity mechanisms for security, high availability and connection pooling, TOCF (EE) also facilitates connectivity to external systems from T24.

JMS (Java Message Service) is an API that provides the facility to create, send and read messages. It provides loosely coupled, reliable and asynchronous communication.

JMS Connector provides JMS based connector implementation for Interaction Framework X to interface with T24 using OFS Message-based interface.

Deployment Configuration

1. JMS Resource Configuration required in the Deployment Descriptor

   The following resources references are expected to get configured to use the respective JNDI resources available in the application server:

   - queue/t24OFSQueue – need to get mapped to **OFS Request Queue 's JNDI resource**.
   - queue/t24OFSReplyQueue – need to get mapped to **OFS Response Queue's JNDI Resource**.
   - jms/jmsConnectionFactory – need to get mapped to **JMS Connection factory Resource**.

2. Application Context - Spring bean configuration

   Below configuration is required in the Application Context.xml to load JMS Based connector factory.

```
<bean id="t24JmsConnectionProperties" class="com.temenos.irf.config.StandardPropertyReader">
<property name="path" value="irf-config/jms.properties"/>
</bean>

<bean id="t24JMSConnectionFactory" class="com.temenos.irf.comms.jms.JMSConnectorFactory">
<property name="propertyReader" ref="t24JmsConnectionProperties"/>
</bean>
```

   irf-config/jms.properties — Contains the following set of configurations to handle retry in case of JMS Queue or Connection related failures:

   - RetryWait=60
   - RetryCount=3
   - ConnectionTimeout=30

3. Remote Connectivity

   This section is applicable only when Interaction Framework components and TAFJ application are deployed in different Servers (specifically JBOSS).

   - Application context configuration

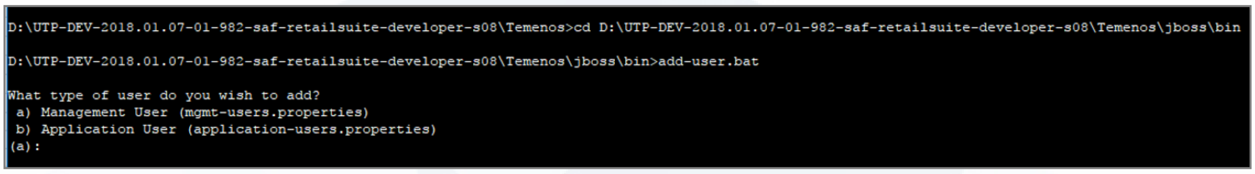
     Application Context configuration in applicationContext.xml will be

```
<bean id="t24ConnectionProperties" class-
s="com.temenos.irf.config.StandardPropertyReader">

<property name="path" value="../irf-config/standalone-comms remote.properties"/>

</bean>

<bean id="serviceLocatorProperties" class-
s="com.temenos.irf.config.StandardPropertyReader">

<property name="path" value="../irf-config/service-locator.properties" />

</bean>
```

- JBoss Management User Creation

  To access TAFJ or TOCFEE JMS Queues, a management user needs to be created in JBOSS ManagementRealm.

  RUN add-user.bat/add-user.sh available in <JBOSS_HOME>/bin directory.



```
D:\UTP-DEV-2018.01.07-01-982-saf-retailsuite-developer-s08\Temenos>cd D:\UTP-DEV-2018.01.07-01-982-saf-retailsuite-developer-s08\Temenos\jboss\bin

D:\UTP-DEV-2018.01.07-01-982-saf-retailsuite-developer-s08\Temenos\jboss\bin>add-user.bat

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a):
```

  The user id and password set using add-user.bat/add-user.sh in the JBoss environment needs to be configured in standalone-comms remote.properties.

- standalone-comms remote.properties

  Sample content of standalone-comms remote.properties is as shown below

```
java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextFactory

java.naming.provider.url=http-remoting://<remote host name>:9089

t24.security.context=INPUTT/xxxxxxx

java.naming.security.principal=mguser1
java.naming.security.credentials=xxxxxxx
```

  java.naming.provider.url — Remote URL of JBoss instance where TAFJ/TOCFEE is deployed

  java.naming.security.principal — Management User created in Remote JBOSS instance

  t24.security.context=INPUTT/xxxxxxx – T24 User to be used to connect T24

  java.naming.security.credentials - Credentials of the Management User (to be saved as plain text)

  internal.reqque.jndiname=jms/queue/t24OFSQueue

  internal.respque.jndiname=jms/queue/t24OFSReplyQueue

  external.reqque.jndiname=jms/queue/t24TCIBQueue

  external.respque.jndiname=jms/queue/t24TCIBReplyQueue

Usage

JMS Connector will be used as a T24 interface connector in Interaction Framework X where

T24 Deployment mode is:

- TAFJ or
- TAFC - where TOCFEE component is enabled.

TCS Connector - TAFC TCS

In earlier releases of T24, Temenos supported T24 Server Services (TSS) which provided a programmable interface for external application to interact with T24 over TCP/IP. The TCS client distribution has necessary libraries to connect to TSS and exchange messages for business transactions. In case of T24 deployment where TSS services are running, TCS connector needs to be enabled for T24 integration.

TCS Connector provides TCS based connector implementation for Interaction Framework X to interface with T24 using OFS Message-based interface.

Deployment Configuration

1. TCS Resource Configuration is required in the Deployment Descriptor

   The following resources references are expected to get configured to use the respective resources available in the Application Server:

   - tcs/tcsConnectionFactory – should get mapped to **TCS Connection factory Resource**.
   - Need to start listeners by invoking the TCServer.bat.

2. Application Context – Spring bean Configuration

   - Below configuration is required in the ApplicationContext.xml to load TCS Based connector factory.

```
<bean id="t24ConnectionProperties" class="com.temenos.irf.config.StandardPropertyReader">
<property name="path" value="../irf-config/tafc-connection.properties"/>
</bean>

<bean id="t24TafcConnectionFactory" class="com.temenos.irf.tcs.TCSConnectionFactory">
<property name="propertyReader" ref="t24ConnectionProperties"/>
</bean>
```

Usage

TCS Connector will be used as a T24 interface connector in Interaction Framework X where T24 Deployment mode is TAFC.

SSL Configuration for IRIS R18 on Jboss 7

This explains you how to setup SSL/HTTPS support for your Jboss 7 server using the pure Java implementation supplied by JSSE.

Firstly, you have to configure keys and (self-signed) certificates for your web server. This guide will briefly explain how to accomplish that.

- **Pure Java SSL-Setup using keytool**

  Generate a secret key/certificate and store it in a file called a "key store". The certificate is valid for 1 year = 365 days. The password use for encryption is "temenos".

  1. **Step 1: Generate key**

     $ keytool -genkey -alias temenos -keyalg RSA -keystore temenos.keystore -validity 365 (***open a cmd prompt pointing to the bin directory eg: ..\jdk8\bin & execute the given command to generate the key***)

     Command Line

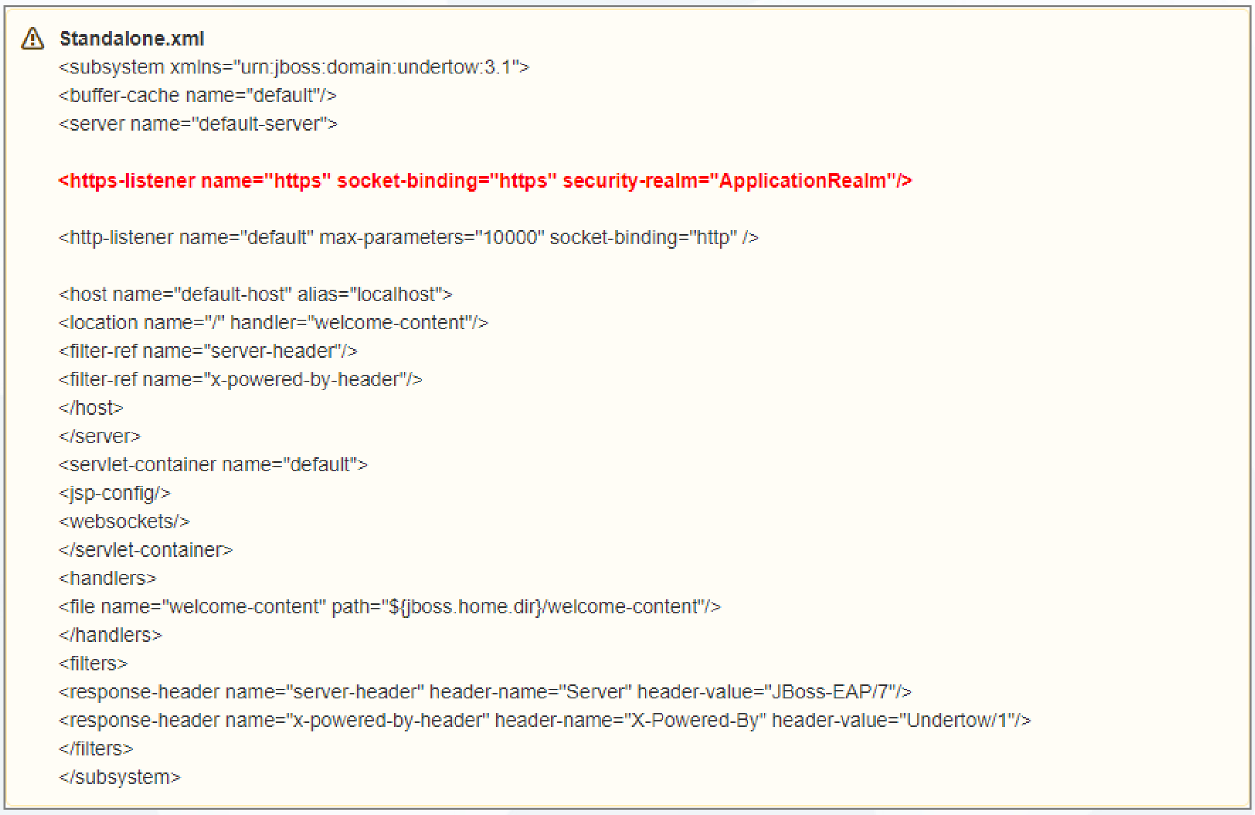
```
Enter keystore password: temenos

Re-enter new password: temenos

What is your first and last name?

[Unknown]: localhost

What is the name of your organizational unit?

[Unknown]: IT

What is the name of your organization?

[Unknown]: temenos

What is the name of your City or Locality?

[Unknown]: Bangalore

What is the name of your State or Province?

[Unknown]: Karnataka

What is the two-letter country code for this unit?

[Unknown]: IN

Is CN=localhost, OU=IT, O=temenos, L=Bangalore, ST=Karnataka, C=IN correct?

[no]: yes
```

```
Enter key password for <deva> temenos

(RETURN if same as keystore password):

Re-enter new password: temenos
```

You will find a key generated "temenos.keystore" in the same jdk bin directory.

2. **Step 2: Configure JBoss 7**

   Once the key is generated, move the key to the configuration directory of your application server(in this case \jboss\stan-dalone\configuration\).Inthestandalone.xmlconfigfile,underthesubsystem<subsystem xmlns-s="urn:jboss:domain:undertow:3.1">, include a http-listener for "https" as follows : (include the highlighted line)



Standalone.xml

```
<subsystem xmlns="urn:jboss:domain:undertow:3.1">
<buffer-cache name="default"/>
<server name="default-server">

<https-listener name="https" socket-binding="https" security-realm="ApplicationRealm"/>

<http-listener name="default" max-parameters="10000" socket-binding="http" />

<host name="default-host" alias="localhost">
<location name="/" handler="welcome-content"/>
<filter-ref name="server-header"/>
<filter-ref name="x-powered-by-header"/>
</host>
</server>
<servlet-container name="default">
<jsp-config/>
<websockets/>
</servlet-container>
<handlers>
<file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
</handlers>
<filters>
<response-header name="server-header" header-name="Server" header-value="JBoss-EAP/7"/>
<response-header name="x-powered-by-header" header-name="X-Powered-By" header-value="Undertow/1"/>
</filters>
</subsystem>
```

3. **<server-identities> in ApplicationRealm realm Definition**

   The server identities section of a realm definition, which is used to define how a server appears to the outside world, currently this element can be used to configure a password to be used when establishing a remote outbound connection.

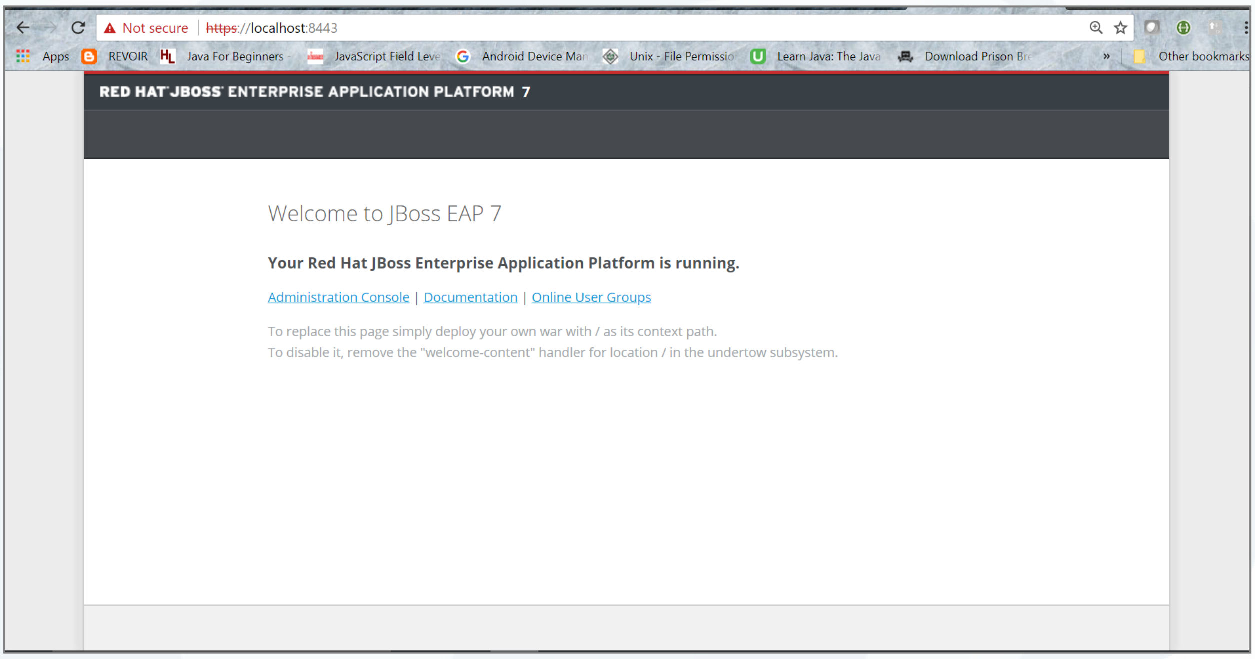
   Add the Highlighted lines:

⚠ **Standalone.xml**

```
<security-realm name="ApplicationRealm">
<server-identities>
<ssl>
<keystore path="temenos.keystore" relative-to="jboss.server.config.dir" keystore-password="temenos" alias="temenos"
key-password="temenos" />
</ssl>
</server-identities>
<authentication>
<local default-user="$local" allowed-users="*" skip-group-loading="true"/>
<properties path="application-users.properties" relative-to="jboss.server.config.dir"/>
</authentication>
<authorization>
<properties path="application-roles.properties" relative-to="jboss.server.config.dir"/>
</authorization>
</security-realm>
```

Once this is done, startup you jboss, by default jboss https runs on port 8443, you can configure as per your wish.

Ensure your jboss is up and running.Try to access https://localhost:8443/ , if you see the below page, its configured properly and listening on configured port 8443.



RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7

Welcome to JBoss EAP 7

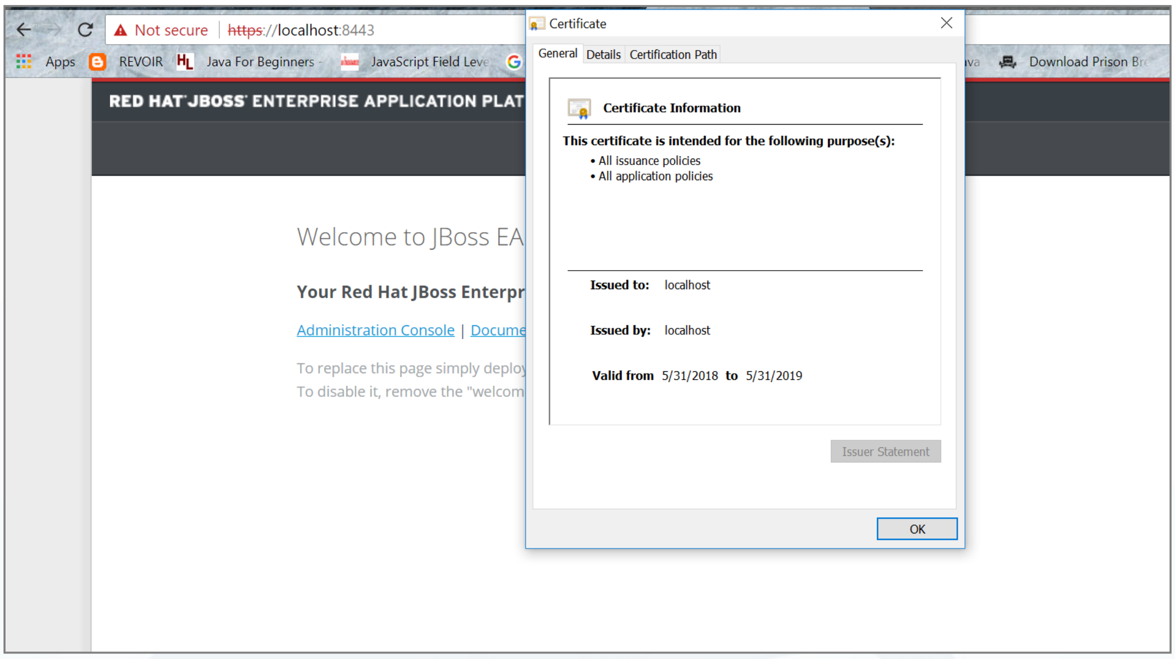
Your Red Hat JBoss Enterprise Application Platform is running.

Administration Console | Documentation | Online User Groups

To replace this page simply deploy your own war with / as its context path.
To disable it, remove the "welcome-content" handler for location / in the undertow subsystem.

4. **Step 4: Certificate Installation in Browser**

- In this step, you need to install the self-signed certificate as per the screen shots below.



Welcome to JBoss EA

Your Red Hat JBoss Enterpr

Administration Console | Docume

To replace this page simply deploy

To disable it, remove the "welcom

Certificate Information

This certificate is intended for the following purpose(s):

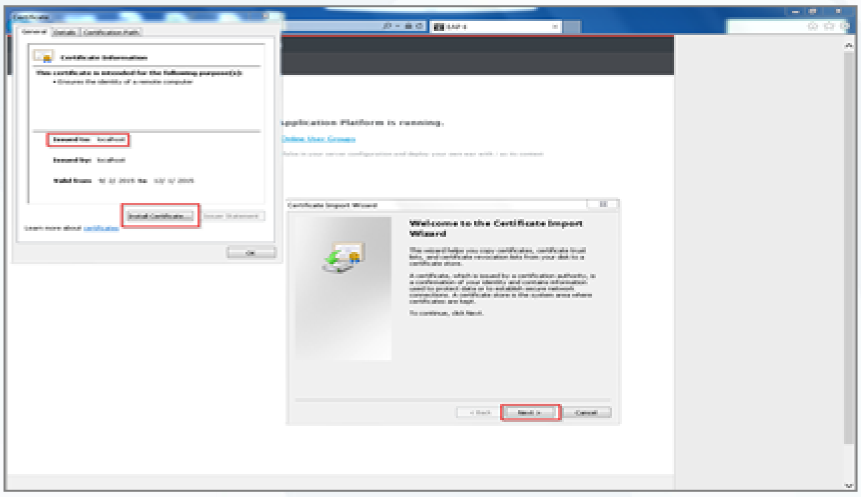
- All issuance policies
- All application policies

Issued to: localhost

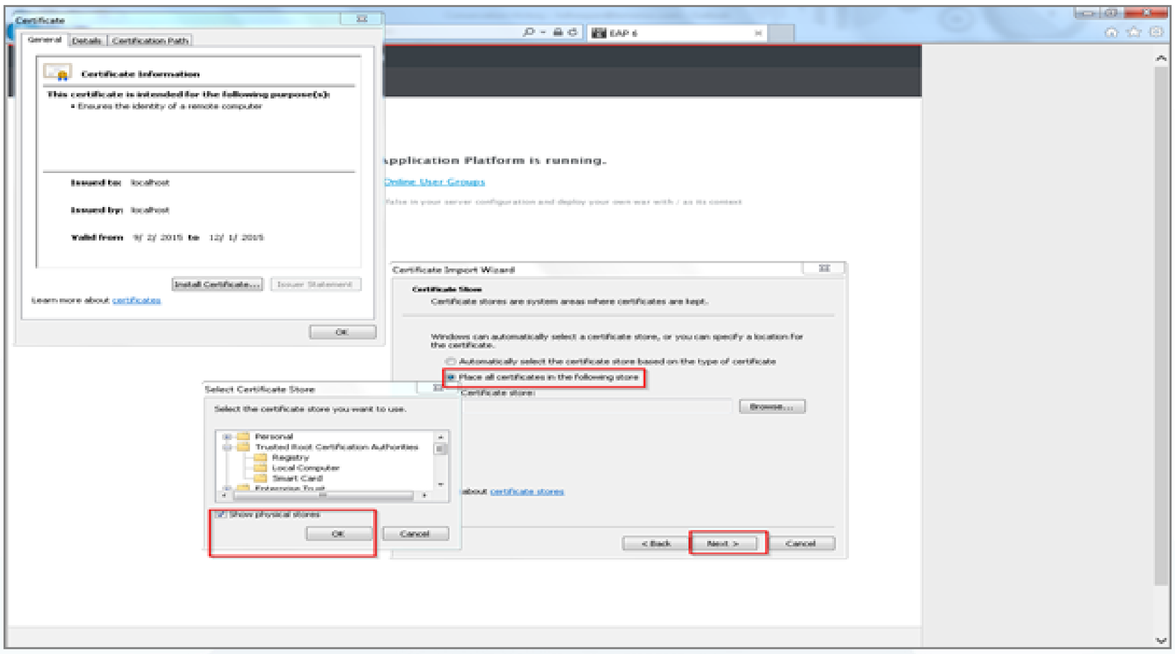
Issued by: localhost

Valid from 5/31/2018 **to** 5/31/2019

- Certificate Installation Wizard



- Provide the path to import the self-signed certificate


Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):
• Ensures the identity of a remote computer

Issued to: localhost

Issued by: localhost

Valid from 9/ 2/ 2015 to 12/ 1/ 2015

Install Certificate... | Issuer Statement

Learn more about certificates

OK

Application Platform is running.

Online User Groups

Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

Automatically select the certificate store based on the type of certificate

Place all certificates in the following store

Certificate store:

Browse...

about certificate stores

< Back | Next > | Cancel

Select Certificate Store

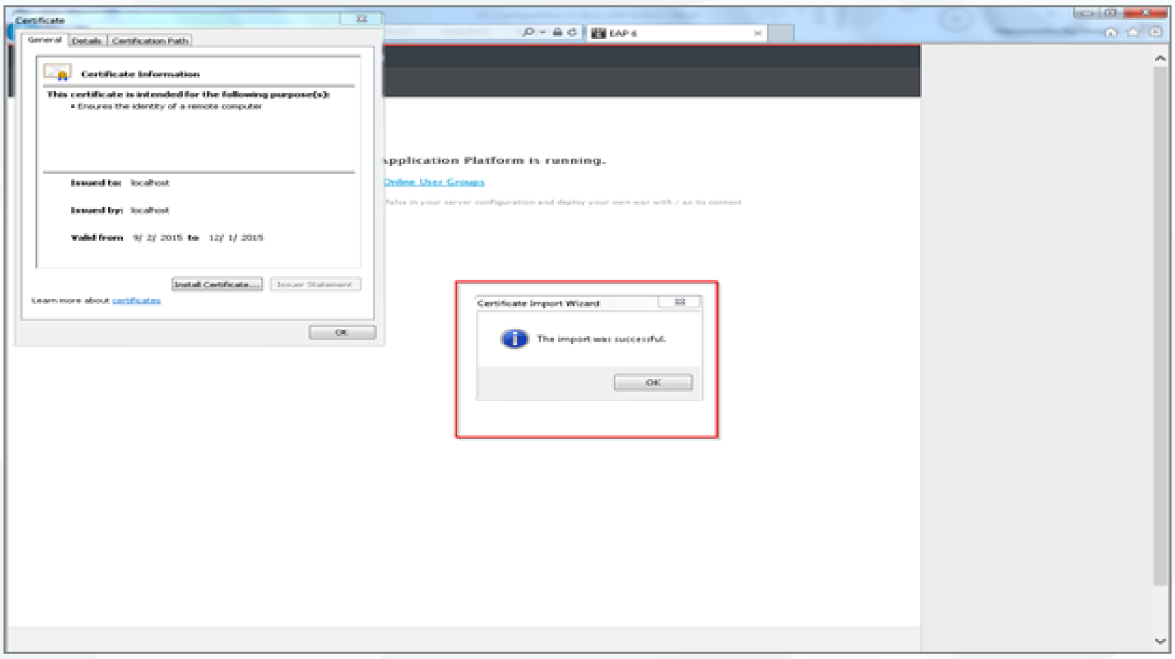
Select the certificate store you want to use.

Personal
Trusted Root Certification Authorities
Registry
Local Computer
Smart Card
Enterprise Trust

Show physical stores

OK | Cancel

- Certificate imported successfully


Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):
• Ensures the identity of a remote computer

Issued to: localhost

Issued by: localhost

Valid from 9/ 2/ 2015 to 12/ 1/ 2015

Install Certificate... | Issuer Statement

Learn more about certificates

OK

Application Platform is running.

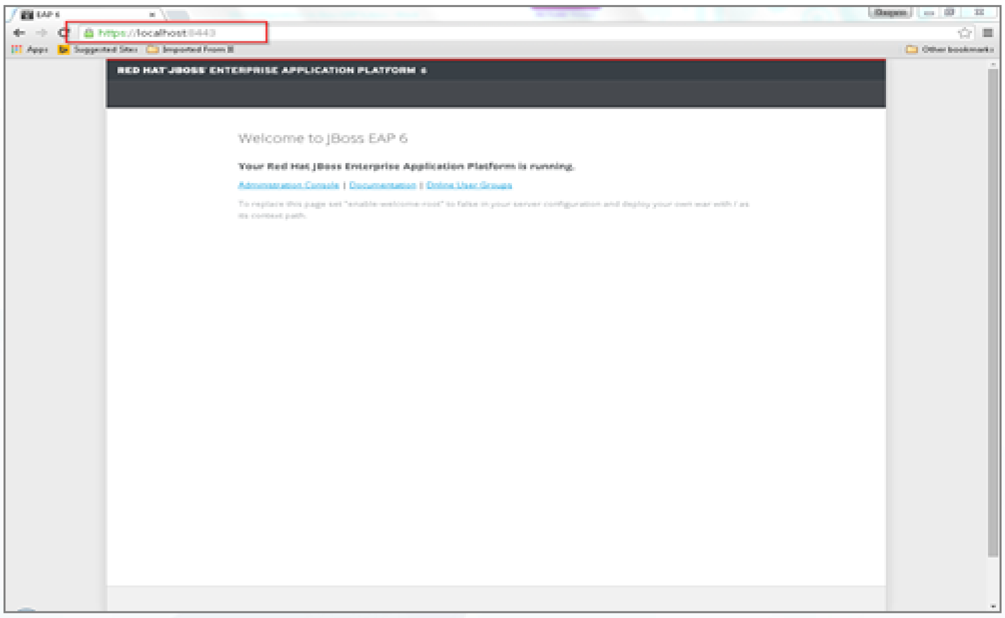
Online User Groups

Certificate Import Wizard

The import was successful.

OK

- Now the SSL works fine with HTTPS



Welcome to JBoss EAP 6

Your Red Hat JBoss Enterprise Application Platform is running.

Administration Console | Documentation | Online User Groups

To replace this page set "enable-welcome-root" to false in your server configuration and deploy your own war with / as its context path.

Note: The name 'localhost' should be same with the Common Name (CN) provided while creating a keystore and truststore.

IRIS R18 Dynamic Mocking

Overview

IRIS Mock Services support dynamic mock response generation based on the fields present as a part of API metadata and dynamically generated data based on the local mock dictionary maintained. IRIS Mock Services are responsible to generate random data for the metadata fields based on the domain mock files maintained for the same. If the fields are not present in the specified domain (specific mock file), it generates random data based on the corresponding field datatype.

Design

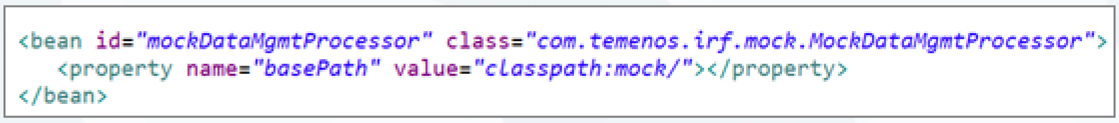
MockDataMgmtProcessor & MockResponder are the classes responsible for handling and generating random data for a request dynamically at run time.

The **MockDataMgmtProcessor** provides implementation to insert, update, view and delete entries into domain specific mock file maintained at basepath specified as a part of **mockDataMgmtProcessor** bean property.

The **MockResponder** provides implementation to generate mock responses based on metadata at run time , it gets corresponding random data by performing a lookup on the domain specific mock file maintained at basepath specified as a part of **mockResponder** bean property, and generating response schema out of it. If the field is not present in the specified mock domain dictionary file, based on the mentioned datatype, it is expected to generate some random data for it.

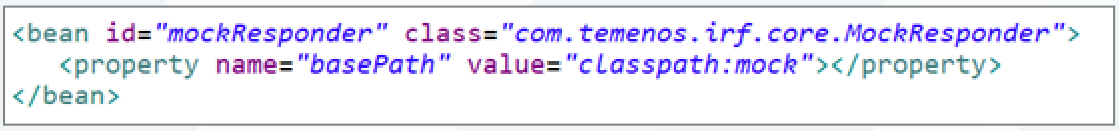
Spring Configurations

The **mockDataMgmtProcessor** bean should be properly configured with basePath property set to the base Directory where mock Files folder holding all the domain specific mock files are placed.



```
<bean id="mockDataMgmtProcessor" class="com.temenos.irf.mock.MockDataMgmtProcessor">
   <property name="basePath" value="classpath:mock/"></property>
</bean>
```

The **mockResponder** bean should be properly configured with basePath property set to the base Directory where mockFiles folder holding all the domain specific mock files are placed.

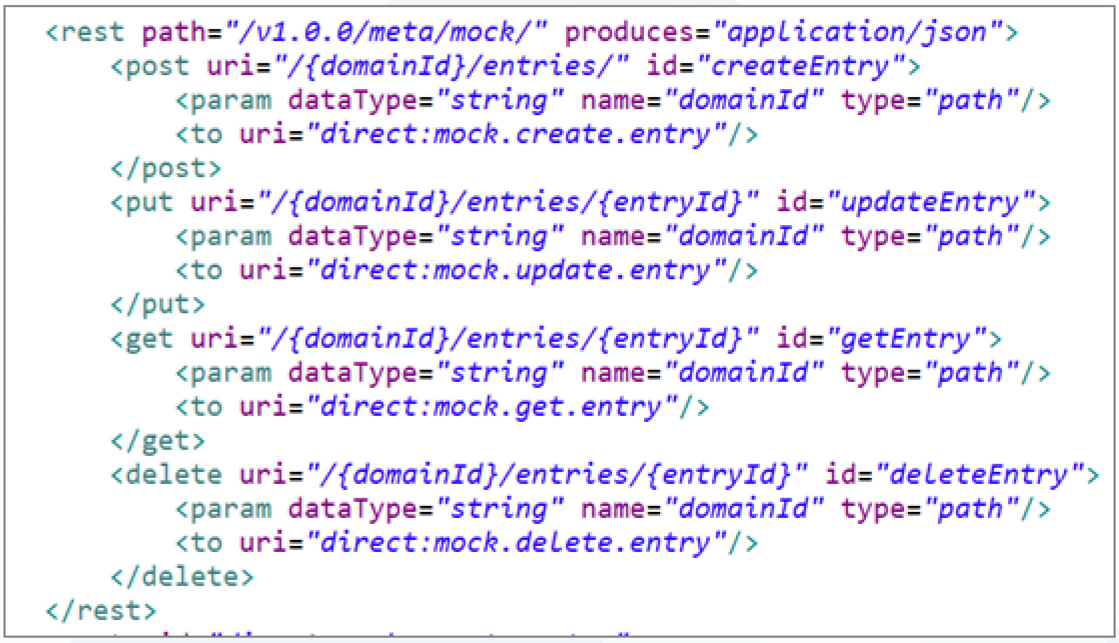


```
<bean id="mockResponder" class="com.temenos.irf.core.MockResponder">
   <property name="basePath" value="classpath:mock"></property>
</bean>
```

Services for Domain Specific Mock Dictionary

There is a provision to perform various operations such as to insert, update, view and delete new entries into the maintained mock dictionary via various services. Corresponding Swagger schema is also published in reference to that.

Note: If you are referring to the mock dictionary from the classpath, then you will not be allowed to add, update or delete any of the mock dictionary files from these services, only view access are permitted.

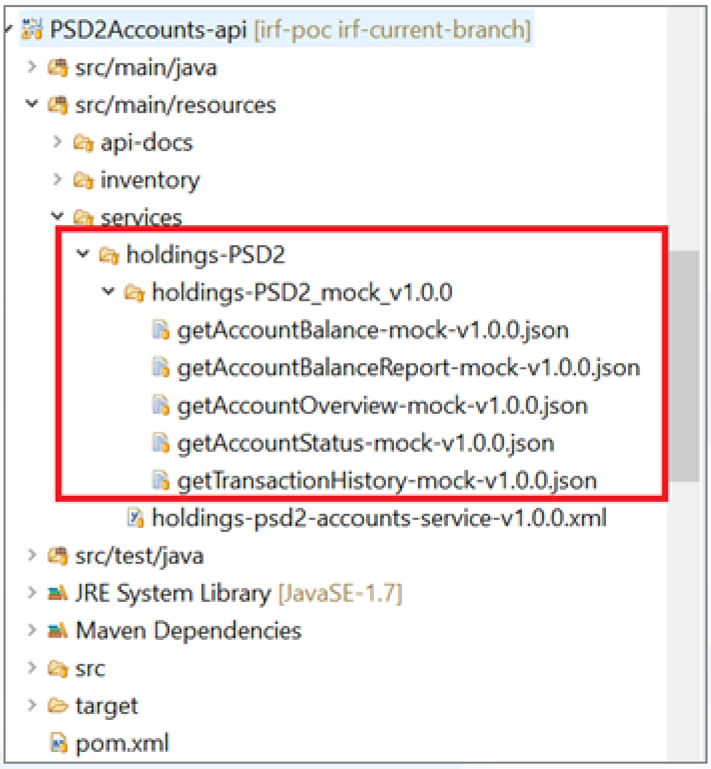


```xml
<rest path="/v1.0.0/meta/mock/" produces="application/json">
    <post uri="/{domainId}/entries/" id="createEntry">
        <param dataType="string" name="domainId" type="path"/>
        <to uri="direct:mock.create.entry"/>
    </post>
    <put uri="/{domainId}/entries/{entryId}" id="updateEntry">
        <param dataType="string" name="domainId" type="path"/>
        <to uri="direct:mock.update.entry"/>
    </put>
    <get uri="/{domainId}/entries/{entryId}" id="getEntry">
        <param dataType="string" name="domainId" type="path"/>
        <to uri="direct:mock.get.entry"/>
    </get>
    <delete uri="/{domainId}/entries/{entryId}" id="deleteEntry">
        <param dataType="string" name="domainId" type="path"/>
        <to uri="direct:mock.delete.entry"/>
    </delete>
</rest>
```

Understanding MockResponder Service

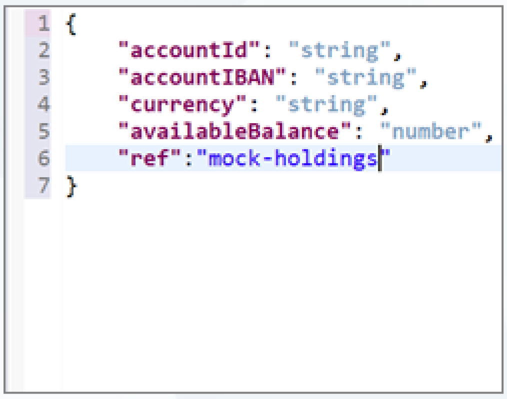
The **MockResponder** provides implementation to generate mock responses based on metadata at run time, it gets corresponding random data by performing a lookup on the domain specific mock file maintained at basepath specified as a part of **mockResponder** bean property, and generating response schema out of it. If the field is not present in the specified mock domain dictionary file, based on the mentioned datatype, it is expected to generate some random data for it.

As a part of APIs created from the Workbench ,you will have a mock json file corresponding to the service generated in a mock directory inside the service directory as shown below:



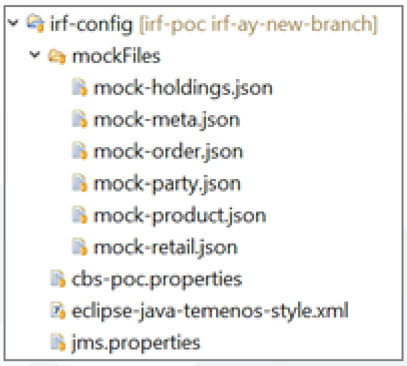
```
PSD2Accounts-api [irf-poc irf-current-branch]
  src/main/java
  src/main/resources
    api-docs
    inventory
    services
      holdings-PSD2
        holdings-PSD2_mock_v1.0.0
          getAccountBalance-mock-v1.0.0.json
          getAccountBalanceReport-mock-v1.0.0.json
          getAccountOverview-mock-v1.0.0.json
          getAccountStatus-mock-v1.0.0.json
          getTransactionHistory-mock-v1.0.0.json
        holdings-psd2-accounts-service-v1.0.0.xml
  src/test/java
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  src
  target
  pom.xml
```

1. Each of the generated mock json contains all the metadata fields along with corresponding datatypes.



```
{
    "accountId": "string",
    "accountIBAN": "string",
    "currency": "string",
    "availableBalance": "number",
    "ref":"mock-holdings"
}
```

2. Along with this, there is a unique field “ref” along with the mapped domain specific mock file, created during generation itself as shown in above figure.
3. During Mocking the Request, IRIS R18 looks into the above file mentioned in the “ref” of the mock json to get random data from already maintained domain specific mock file.
4. If “**ref**” is not present in the json, it just generates the response same as in the generated mock json.

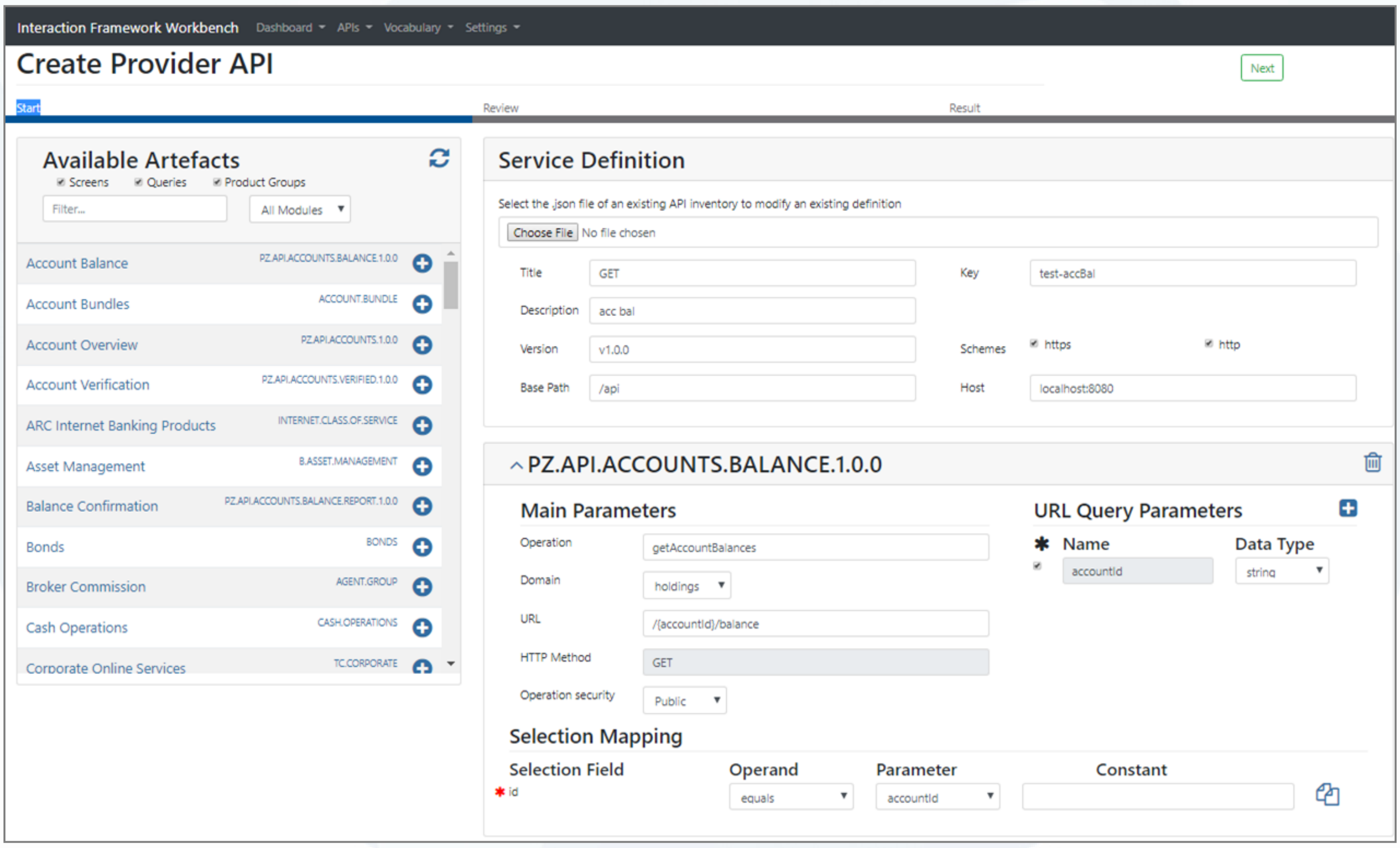


```
irf-config [irf-poc irf-ay-new-branch]
  mockFiles
    mock-holdings.json
    mock-meta.json
    mock-order.json
    mock-party.json
    mock-product.json
    mock-retail.json
  cbs-poc.properties
  eclipse-java-temenos-style.xml
  jms.properties
```

5. It perfoms a lookup in the mock dictionary files based on the domain in the generated mock json and get random values for all the fields in the mock json.
6. If not present, it randomly generates values based on the datatypes of the fields mentioned in the generated mock json.

Steps you need to Follow to Mock your Service

Step 1 : Create a new service



Create Provider API

Available Artefacts

Screens ✓ Queries ✓ Product Groups ✓

Filter...
All Modules

Account Balance	PZ.API.ACCOUNTS.BALANCE.1.0.0
Account Bundles	ACCOUNT.BUNDLE
Account Overview	PZ.API.ACCOUNTS.1.0.0
Account Verification	PZ.API.ACCOUNTS.VERIFIED.1.0.0
ARC Internet Banking Products	INTERNET.CLASS.OF.SERVICE
Asset Management	B.ASSET.MANAGEMENT
Balance Confirmation	PZ.API.ACCOUNTS.BALANCE.REPORT.1.0.0
Bonds	BONDS
Broker Commission	AGENT.GROUP
Cash Operations	CASH.OPERATIONS
Corporate Online Services	TC.CORPORATE

Service Definition

Select the .json file of an existing API inventory to modify an existing definition

Choose File No file chosen

Title	GET	Key	test-accBal
Description	acc bal		
Version	v1.0.0	Schemes	✓ https ✓ http
Base Path	/api	Host	localhost:8080

PZ.API.ACCOUNTS.BALANCE.1.0.0

Main Parameters

Operation	getAccountBalances
Domain	holdings
URL	/{accountId}/balance
HTTP Method	GET
Operation security	Public

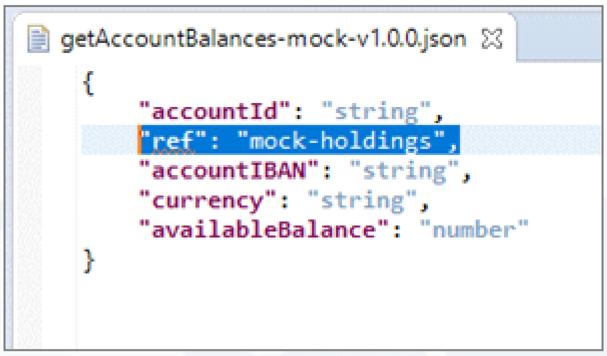
URL Query Parameters

*	Name	Data Type
✓	accountId	string

Selection Mapping

Selection Field	Operand	Parameter	Constant
* id	equals	accountId	

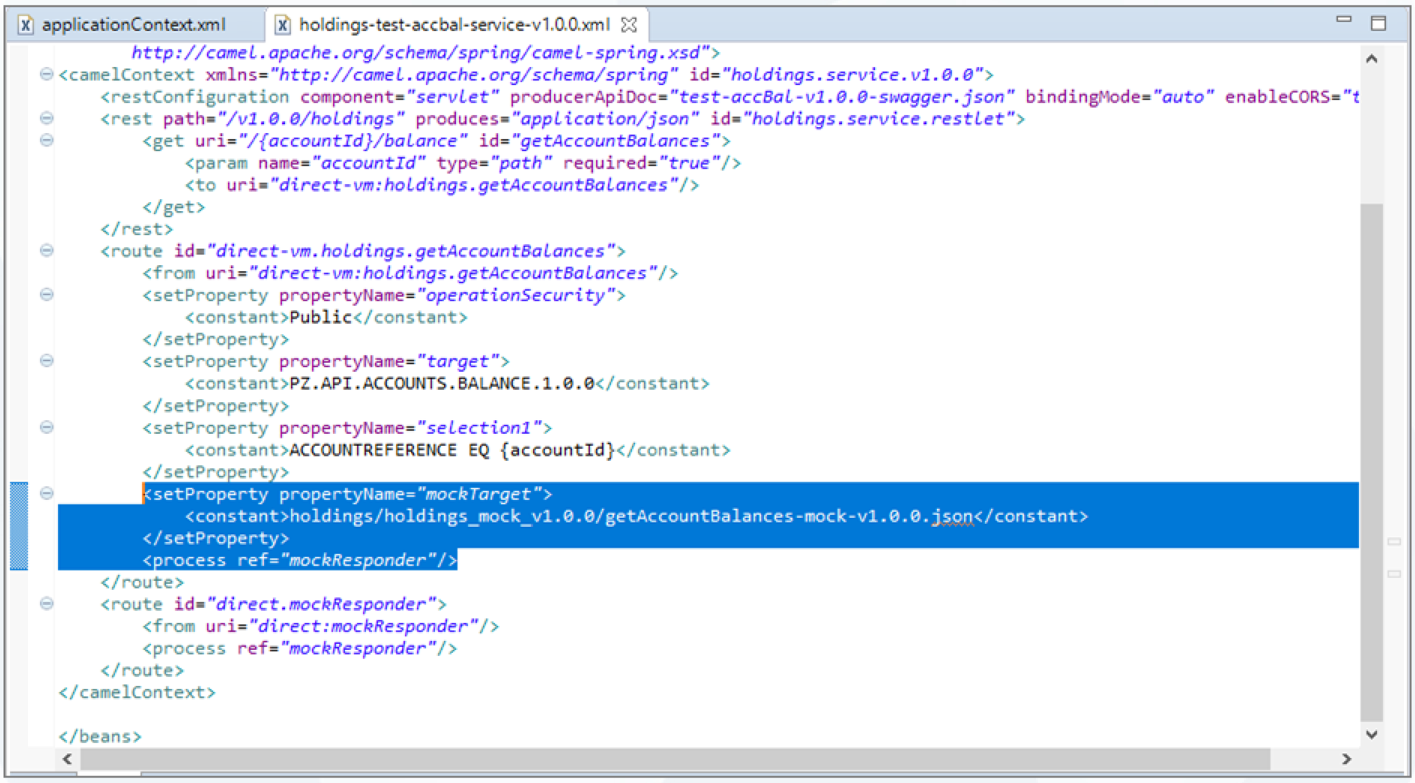
Step 2 : Once Service is generated, Check the generated mock file for "ref" in the json is proper or not.



getAccountBalances-mock-v1.0.0.json

```json
{
    "accountId": "string",
    "ref": "mock-holdings",
    "accountIBAN": "string",
    "currency": "string",
    "availableBalance": "number"
}
```

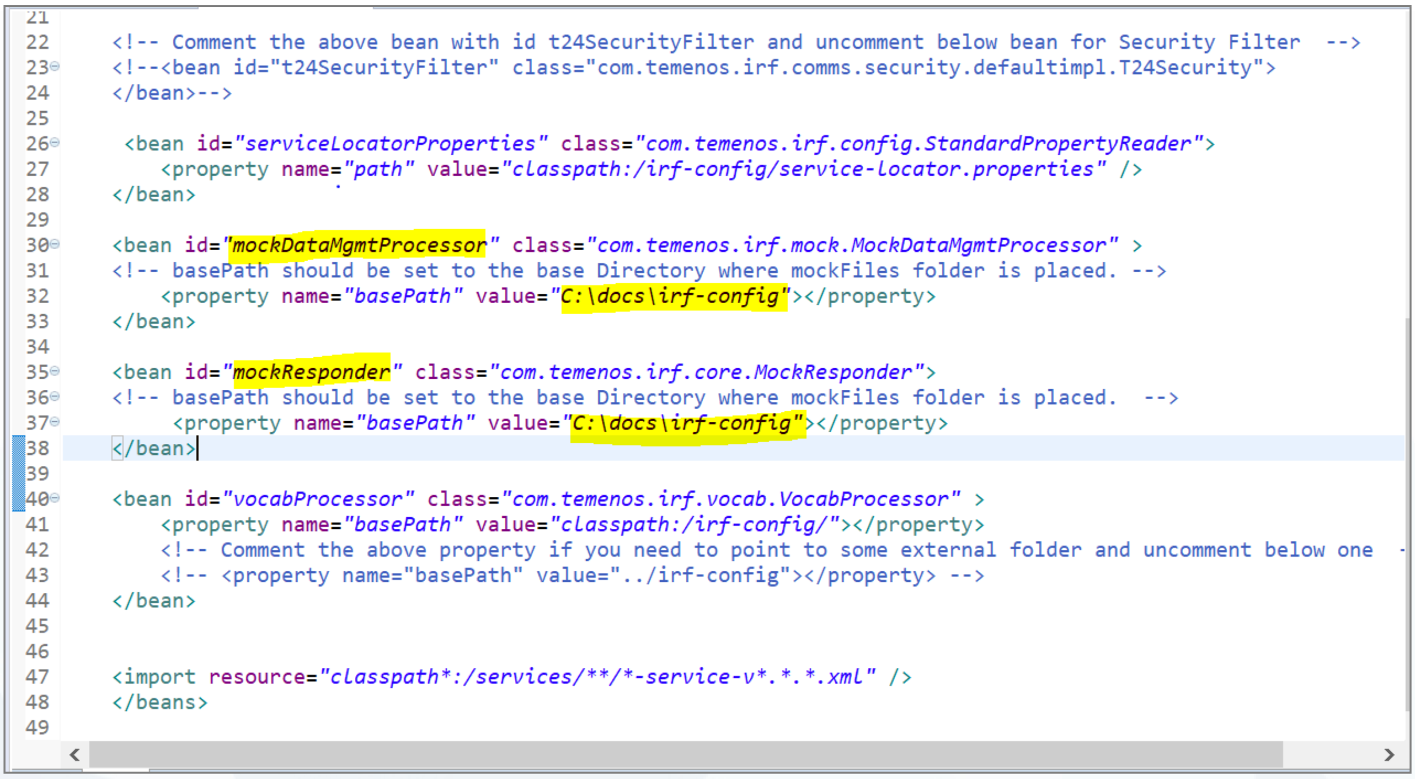
Step 3 : Modify the service xml, configure the property to set "mockTarget" pointing it to the generated mock file for reference to fetch metadata and "ref" for dictionary reference at run-time.



applicationContext.xml | holdings-test-accbal-service-v1.0.0.xml

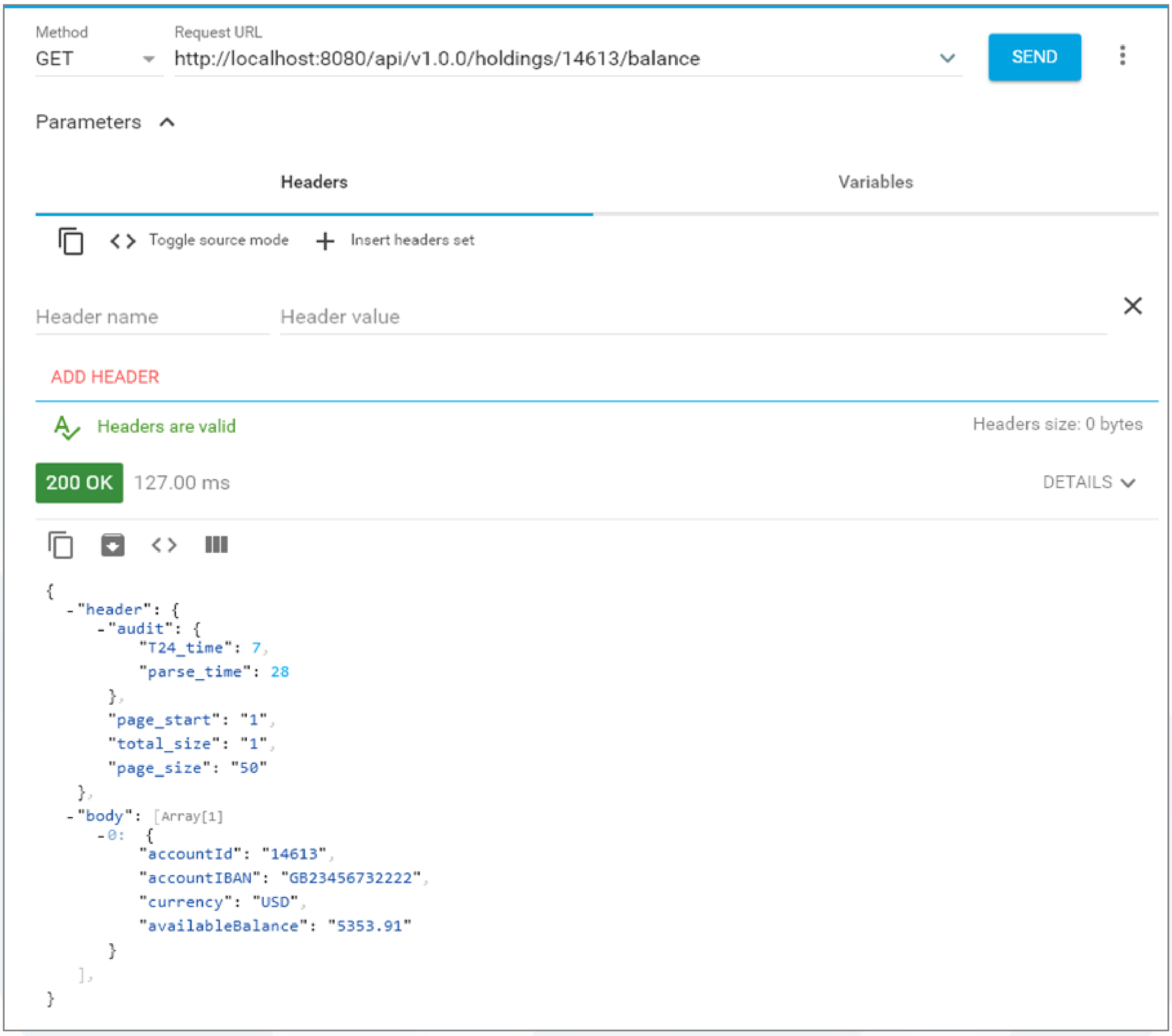
```xml
        http://camel.apache.org/schema/spring/camel-spring.xsd">
<camelContext xmlns="http://camel.apache.org/schema/spring" id="holdings.service.v1.0.0">
    <restConfiguration component="servlet" producerApiDoc="test-accBal-v1.0.0-swagger.json" bindingMode="auto" enableCORS="t
    <rest path="/v1.0.0/holdings" produces="application/json" id="holdings.service.restlet">
        <get uri="/{accountId}/balance" id="getAccountBalances">
            <param name="accountId" type="path" required="true"/>
            <to uri="direct-vm:holdings.getAccountBalances"/>
        </get>
    </rest>
    <route id="direct-vm.holdings.getAccountBalances">
        <from uri="direct-vm:holdings.getAccountBalances"/>
        <setProperty propertyName="operationSecurity">
            <constant>Public</constant>
        </setProperty>
        <setProperty propertyName="target">
            <constant>PZ.API.ACCOUNTS.BALANCE.1.0.0</constant>
        </setProperty>
        <setProperty propertyName="selection1">
            <constant>ACCOUNTREFERENCE EQ {accountId}</constant>
        </setProperty>
        <setProperty propertyName="mockTarget">
            <constant>holdings/holdings_mock_v1.0.0/getAccountBalances-mock-v1.0.0.json</constant>
        </setProperty>
        <process ref="mockResponder"/>
    </route>
    <route id="direct.mockResponder">
        <from uri="direct:mockResponder"/>
        <process ref="mockResponder"/>
    </route>
</camelContext>

</beans>
```

Step 4 : Ensure that the mockResponder and mockDataMgmtProcessor beans are configured properly in ApplicationContext.xml so that the mock dictionary can be reference to generate mock Data.



```
    <!-- Comment the above bean with id t24SecurityFilter and uncomment below bean for Security Filter  -->
    <!--<bean id="t24SecurityFilter" class="com.temenos.irf.comms.security.defaultimpl.T24Security">
    </bean>-->

     <bean id="serviceLocatorProperties" class="com.temenos.irf.config.StandardPropertyReader">
        <property name="path" value="classpath:/irf-config/service-locator.properties" />
    </bean>

    <bean id="mockDataMgmtProcessor" class="com.temenos.irf.mock.MockDataMgmtProcessor" >
    <!-- basePath should be set to the base Directory where mockFiles folder is placed. -->
        <property name="basePath" value="C:\docs\irf-config"></property>
    </bean>

    <bean id="mockResponder" class="com.temenos.irf.core.MockResponder">
    <!-- basePath should be set to the base Directory where mockFiles folder is placed.  -->
         <property name="basePath" value="C:\docs\irf-config"></property>
    </bean>

    <bean id="vocabProcessor" class="com.temenos.irf.vocab.VocabProcessor" >
        <property name="basePath" value="classpath:/irf-config/"></property>
        <!-- Comment the above property if you need to point to some external folder and uncomment below one  -
        <!-- <property name="basePath" value="../irf-config"></property> -->
    </bean>


    <import resource="classpath*:/services/**/*-service-v*.*.*.xml" />
    </beans>

```

Step 5 : Add the service dependency in the container project, start the jetty and you are good to go. Test your Service.



```
http://localhost:8080/api/v1.0.0/holdings/14613/balance
```

200 OK 127.00 ms

```
{
  -"header": {
     -"audit": {
          "T24_time": 7,
          "parse_time": 28
     },
     "page_start": "1",
     "total_size": "1",
     "page_size": "50"
  },
  -"body": [Array[1]
     -0:  {
          "accountId": "14613",
          "accountIBAN": "GB23456732222",
          "currency": "USD",
          "availableBalance": "5353.91"
     }
  ],
}
```

Infrastructure Services

This section covers the following topics.

Directory Services

Directory based service locator is one of the three, lookup implementation for dynamically listing the available APIs and displaying the swagger spec. This service locator implementation is available in irf-service-locator and configurable in Application Context xml file as properties of processor ServiceDirectoryProcessor. IRIS R18 APIs adopts Open API Specification and defines schema for all services supported.

Listing the services

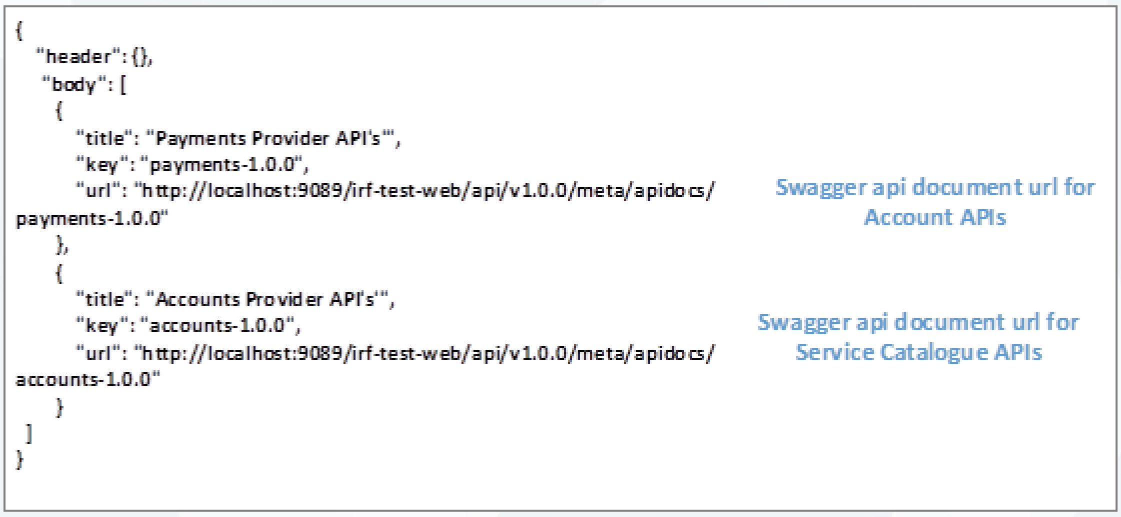
Users can discover the services supported using the discovery url. Below url helps users to get the swagger documentation url for all the services supported in T24 deployment.

http:<base url>/api/v1.0.0/meta/apidocs?

Where base url will be used to access IRIS R18 application.

Example: http://mybank.com/IRIS-Web/api/v1.0.0/meta/apidocs ?

Sample Response:



```
{
  "header": {},
   "body": [
     {
        "title": "Payments Provider API's'",
        "key": "payments-1.0.0",
        "url": "http://localhost:9089/irf-test-web/api/v1.0.0/meta/apidocs/
payments-1.0.0"
     },
     {
        "title": "Accounts Provider API's'",
        "key": "accounts-1.0.0",
        "url": "http://localhost:9089/irf-test-web/api/v1.0.0/meta/apidocs/
accounts-1.0.0"
     }
  ]
}
```

Swagger api document url for Account APIs
Swagger api document url for Service Catalogue APIs

Additionally users can also filter the services by using below query parameters:

- resource- used to filter based on the resource name.

  Example: resource=”account-info”

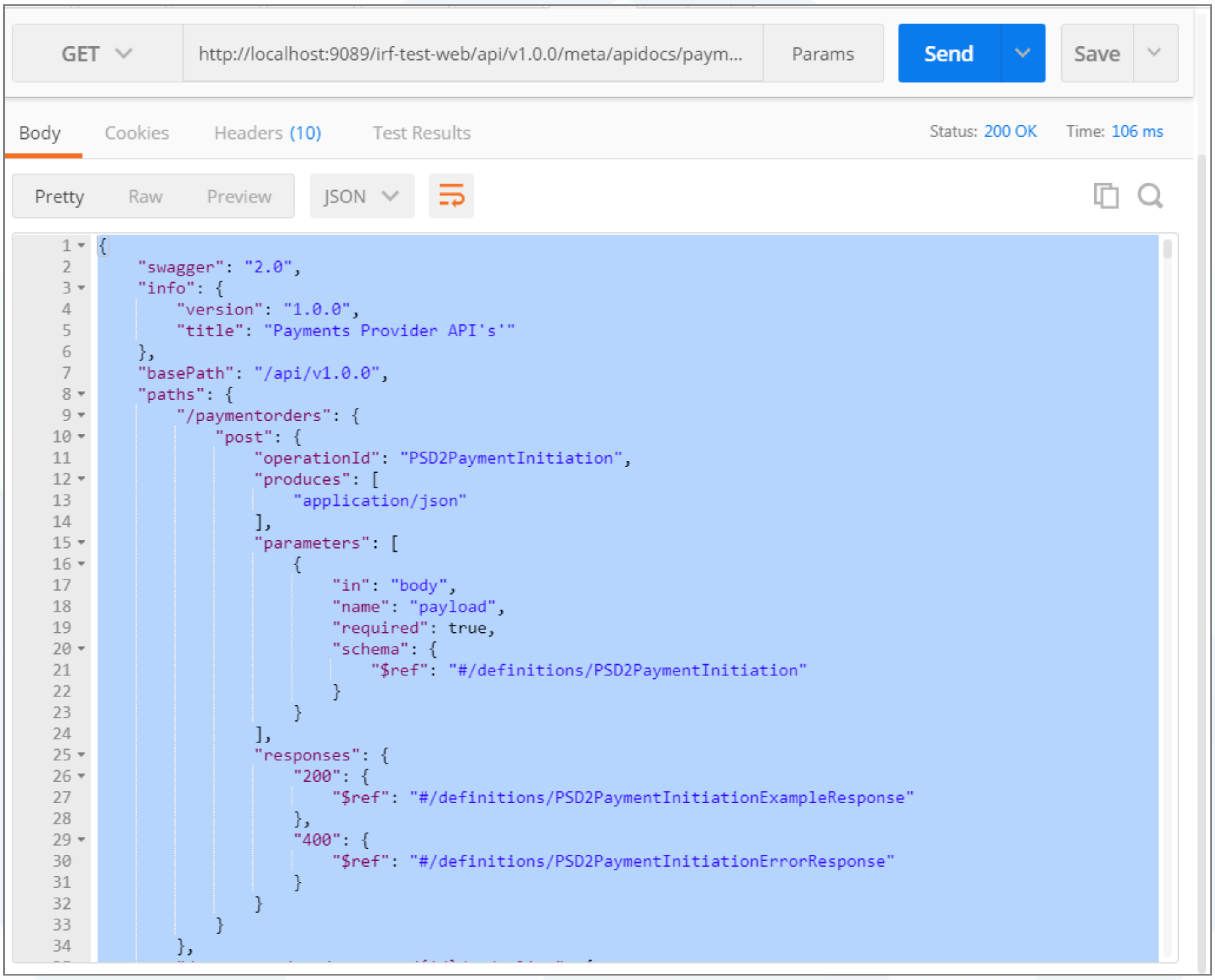
- tag- used to filter based on the tags specified in the swagger API specification.

  Example: tag = “accounts”

Swagger Specification

The url available in the response of **API Docs** request can be used to get the swagger api documentation of the provider apis.

"http://localhost:9089/irf-test-web/api/v1.0.0/meta/apidocs/payments-1.0.0"



```
{
    "swagger": "2.0",
    "info": {
        "version": "1.0.0",
        "title": "Payments Provider API's'"
    },
    "basePath": "/api/v1.0.0",
    "paths": {
        "/paymentorders": {
            "post": {
                "operationId": "PSD2PaymentInitiation",
                "produces": [
                    "application/json"
                ],
                "parameters": [
                    {
                        "in": "body",
                        "name": "payload",
                        "required": true,
                        "schema": {
                            "$ref": "#/definitions/PSD2PaymentInitiation"
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "$ref": "#/definitions/PSD2PaymentInitiationExampleResponse"
                    },
                    "400": {
                        "$ref": "#/definitions/PSD2PaymentInitiationErrorResponse"
                    }
                }
            }
        },
```

Tools

This section covers the following topics.

RIM Importer

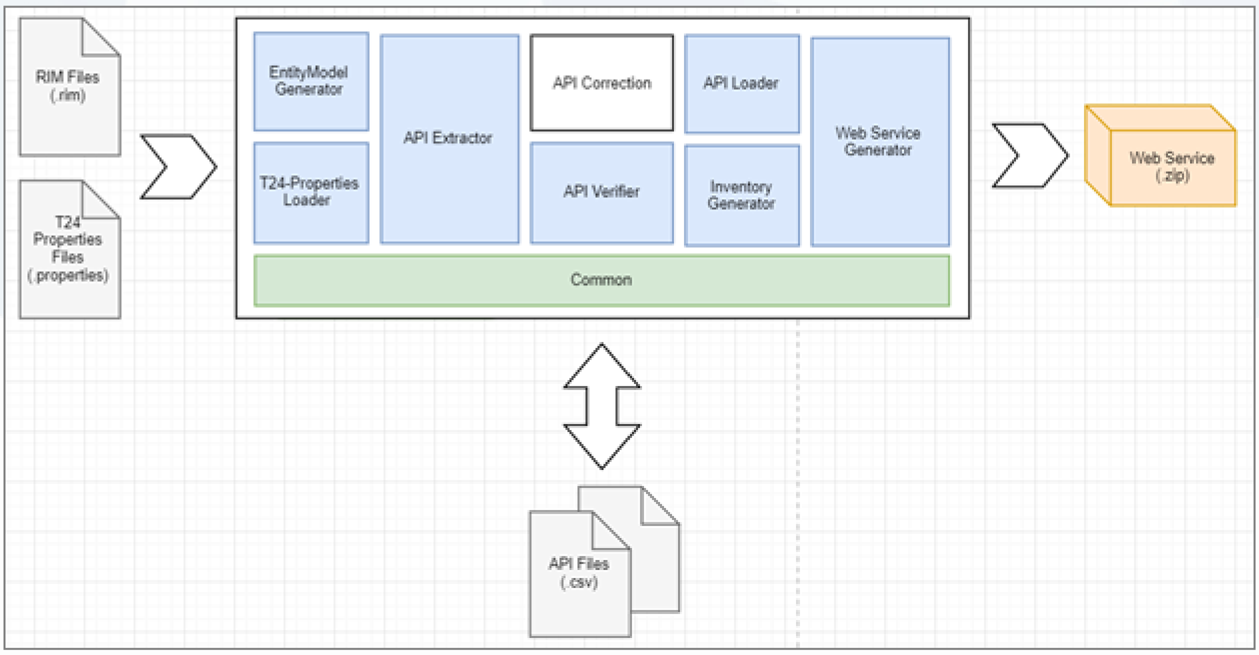
This section covers the following topics.

Overview

RIM Importer is a part of IRIS R18 tool chain, the tool intends to address the challenge of migration of legacy IRIS generated API's to current IRIS R18. The Tool also intends to migrate 80% of the existing API involving both manual and automated work process. The tool helps to "Load, Transform, Validate and Generate" API Services that is compatible to IRIS R18.

Note: The generated API Services is still up to the user or an individual to test and verify for intent.

The IRIS R18 API migration is a commandline based tool that helps in reducing the effort of API migration. Below is the architectural diagram for RIM Importer. RIM Importer intends to take RIM and generated properties files to convert the legacy IRIS generated API to IRIS R18 compliant API as Service i.e CamelContext (.zip) files. The RIM files alone is sufficient if the artefact naming convention was followed which case the generated properties files are optional, though recommended for accurate and reliable extraction of API's and is must for field mapping for the ENQUIRY type of artifacts. The tool generates intermediate API (.csv) files for verification, nice REST compliance of the URL, other details to be compliant to IRIS R18 convention and rules as needed or applicable. The updated API (.csv) files is fed back to tool for API Service generation harnessing the existing IRIS R18 tooling system. The output of the system is Camel Service Context (.zip) files which are consumed in the IRIS R18 Service Component which in turn gets hosted by the IRIS R18 Container getting deployed and hosted on the Web Server as Container to host the T24 artefacts as L2 API Service aka Provider API's.



Workflow

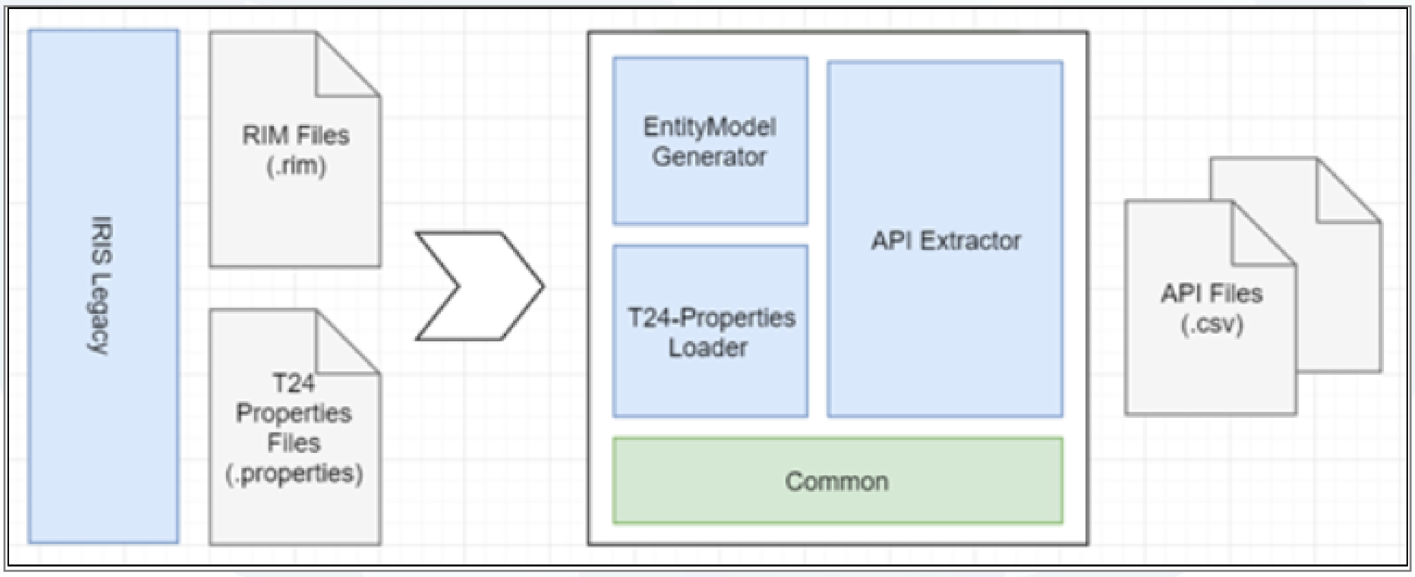
RIM Importer performs migration from Legacy IRIS System to IRIS R18 in three steps which involves both manual and automated work job towards correction and generation.

- API Extraction
- API Validation
- API Generation

API Extraction

The base RIM and T24 Properties Files from the delivered API's from the legacy IRIS is provided for API Extraction project wise. The artifacts are used to reverse engineer the API information and to compile a list of API for the migration. The extracted API list from each of the Project are compiled to respective API CSV files.

Once the API's are extracted from the provided RIMS, the tool can automatically create the relevant artefact in T24 Server by following the IRIS R18 convention making completely discoverable to the IRIS R18 designer to extend and validate the APIs using the designer. The auto creation of the artefact automatically updates resource and field description compatible to vocabulary. Vocabularies are maintained by the vertical domain experts and by the API governance team. Vocabulary standardizes the rest api path formation based on domain terms. The artefact creation step by default is not mandatory but if the naming convention is satisfactory to the expectation then auto creation of the artefact is recommended.



API Extraction

Usage : -e -f C:\IRIS-Importer\TCImages -o C:\IRIS-Importer\TCImages.csv

-e : (M) Generates the Entity Models from RIMS and optionally loads the properties files if input to the importer to extract the list of API's and details.

-f : (M) The input RIMS and Properties files are provided as files.

or

-d : (M) The input RIMS and Properties files are provided as directory.

-p : (O) Input importer properties file e.g C:\IRIS-Importer\importer.properties

-o : (O) Specifies the output file and location where the raw Extracted API is saved. This token is optional, when not specified, gets saved in a file **"IRIS-ExtractedAPIs.csv"** in the current working directory of the IRIS R18 Importer Console Application.

Artefact Creation

Usage : -c -f C:\IRIS-Importer\Release\output.csv -p C:\IRIS-Importer\Release\importer.properties -m C:\IRIS-Importer\Release\domains.properties

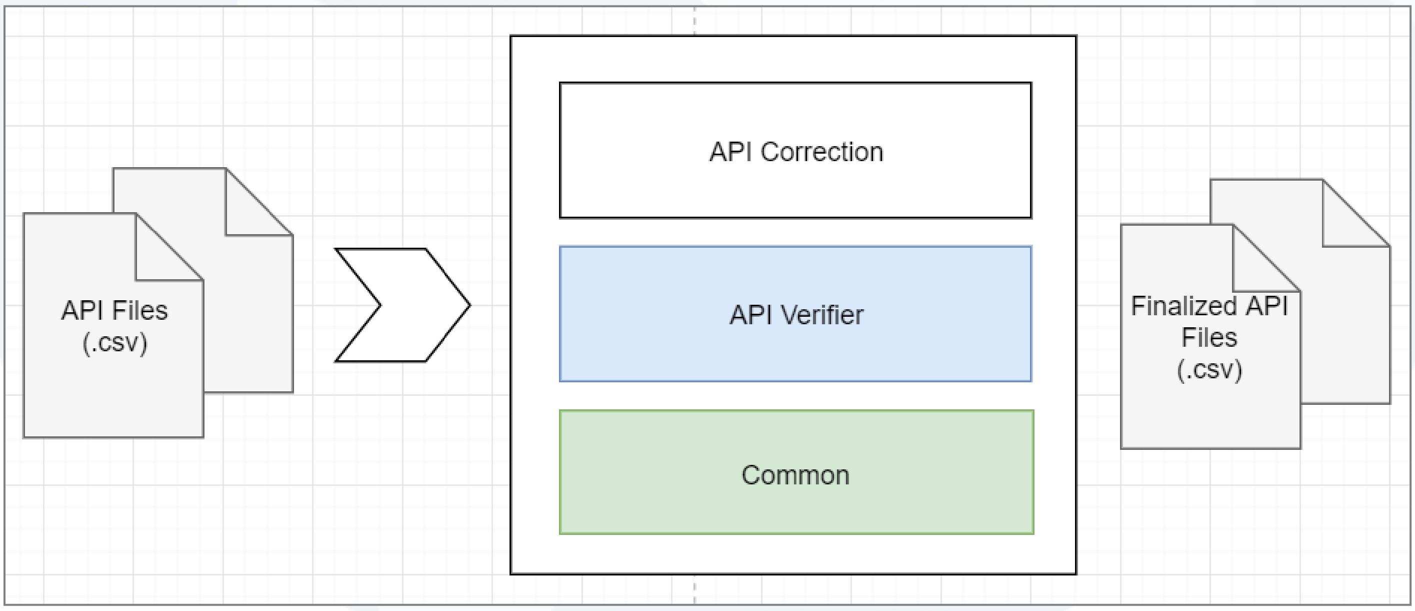
-c : (M) Sets the importer in automatic artefact creation mode, the extracted api's output file is loaded to generate artefact compatible with IRIS R18 API specification from old artefact.

-p : (M) Input importer properties file e.g C:\IRIS-Importer\importer.properties

-m : (M) Specifies the domain properties file that classifies or categorizes the artefact into domain (Maintained by each vertical or by governance team)

API Validation

The extracted API CSV is then audited manually, the URL and parameters for the enquiries are corrected manually as required to be compatible with IRIS R18 specification. The audited files are then sent to T24 for artifact verification and validation upon which the results are saved as finalized API CSV files.



Usage : -v -f C:\IRIS-Importer\TCImages.csv -o C:\IRIS-Importer\TCImagesEx.csv

-v : (M) Sets the importer in validation mode, the extracted list of API's and details are loaded for validation especially URL and Artefacts(VERSION and ENQUIRY).

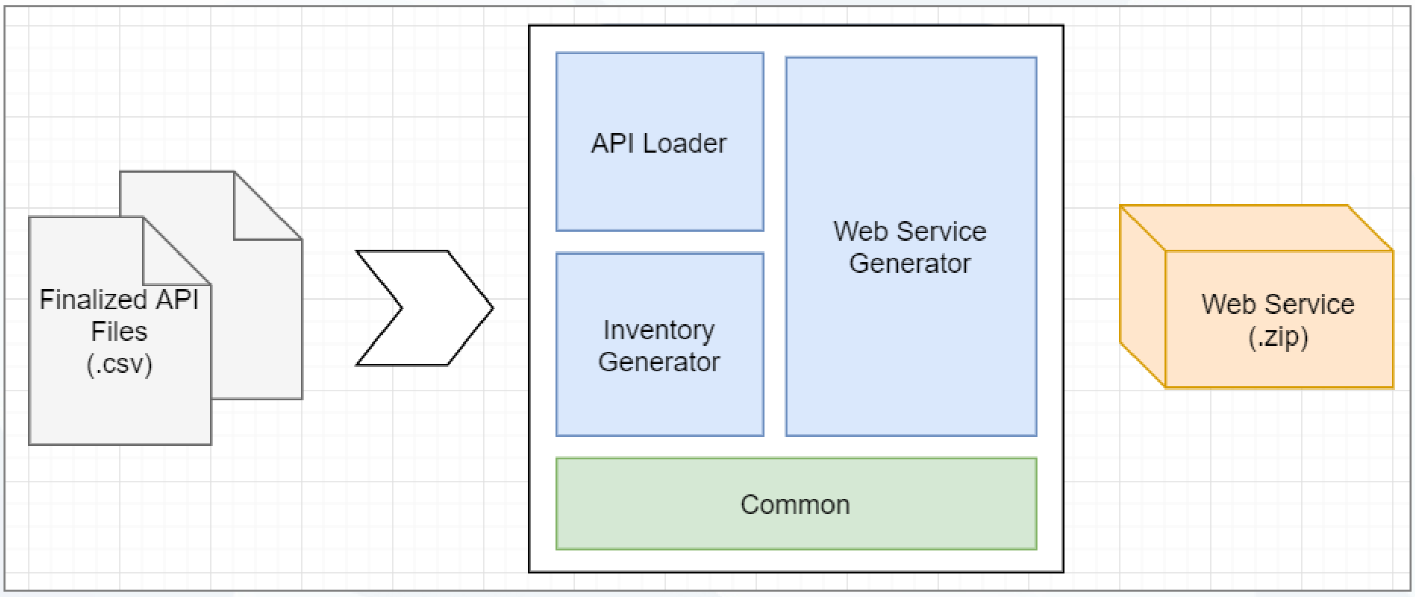
-f : (M) Input extracted api list file e.g C:\IRIS-Importer\TCImage.csv

-p : (O) Input importer properties file e.g C:\IRIS-Importer\importer.properties

-o : (O) Specifies the output file and location where the validated Extracted API is saved, token is optional.

API Generation

The finalized APIs are loaded for an intermediate inventory(json) file generation compatible to IRIS R18 API Web Service generation. The inventory file is then used to generate API Services(.zip) using existing IRIS R18 Tooling.



Inventory Generation

Usage : -i -f C:\IRIS-Importer\TCImagesEx.csv -o C:\IRIS-Importer\Importer-Provider-Inventory.json

-i : (M) Sets the importer in inventory generation mode, the extracted & validated list of API Details are loaded for inventory generation compatible with IRIS R18 API specification.

-f : (M) Input extracted & validated API file e.g C:\IRIS-Importer\TCImagesEx.csv

-o : (O) Specifies the output inventory file and location where the Inventory file is saved, token is optional

Camel Service Context Generation

Usage : -s -f C:\IRIS-Importer\Release\Importer-Provider-Inventory.json -o C:\IRIS-Importer\Release\

-s : (M) Sets the importer in camel service context generation mode, the generated and validated inventory file is loaded for camel service context for the APIs compatible with IRIS R18 API specification.

-f : (M) Input generated API Inventory file e.g C:\IRIS-Importer\Release\Importer-Provider-Inventory.json

-o : (O) Specifies the output path or location where the Services needs to be generated and saved

Getting Started

Prerequisites

1. Java/JDK 1.8

   http://www.oracle.com/ , JAVA_HOME is set appropriately

2. Maven 3.3+

   https://maven.apache.org/ , MAVEN_HOME is set appropriately

3. Latest UTP with TAFJ

   http://utp.temenosgroup.com/ , TAFJ_HOME is set appropriately

4. Eclipse Oxygen or any preferred version

   http://www.eclipse.org/downloads/eclipse-packages/

5. REST Client or POSTMAN or Insomnia (*Optional*)

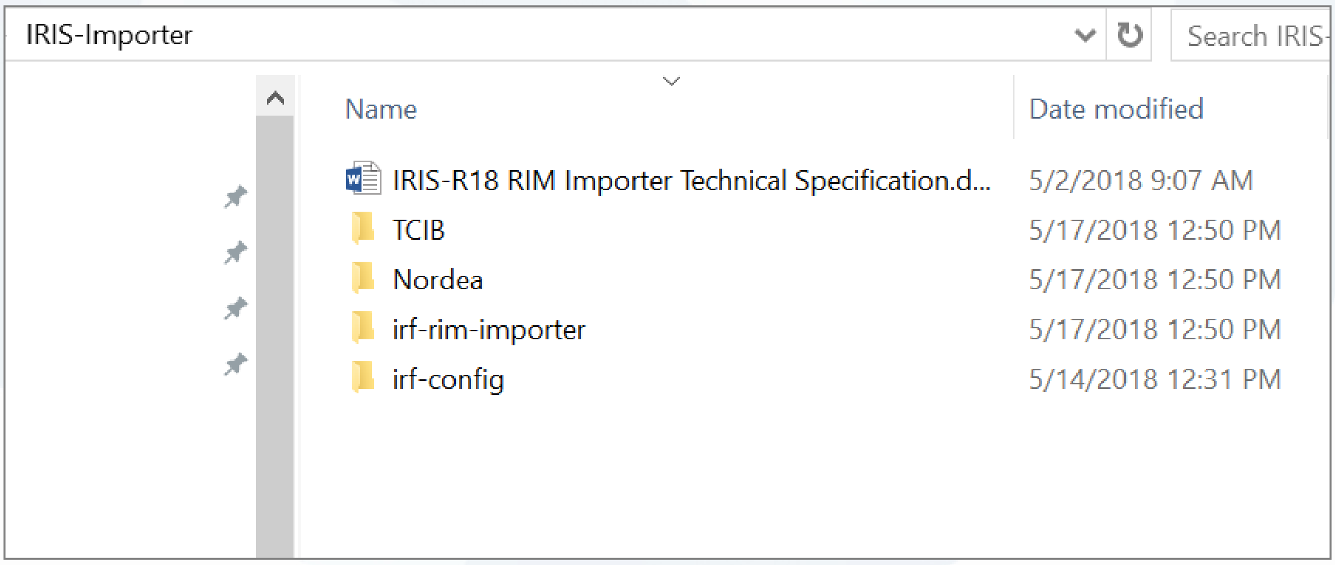
6. Browser Extensions : JSON Formatter, JSONView, etc (*Optional*)

Procedure

1. Download latest rimimporter released pack from Temenos Maven Repository Click here to Download.

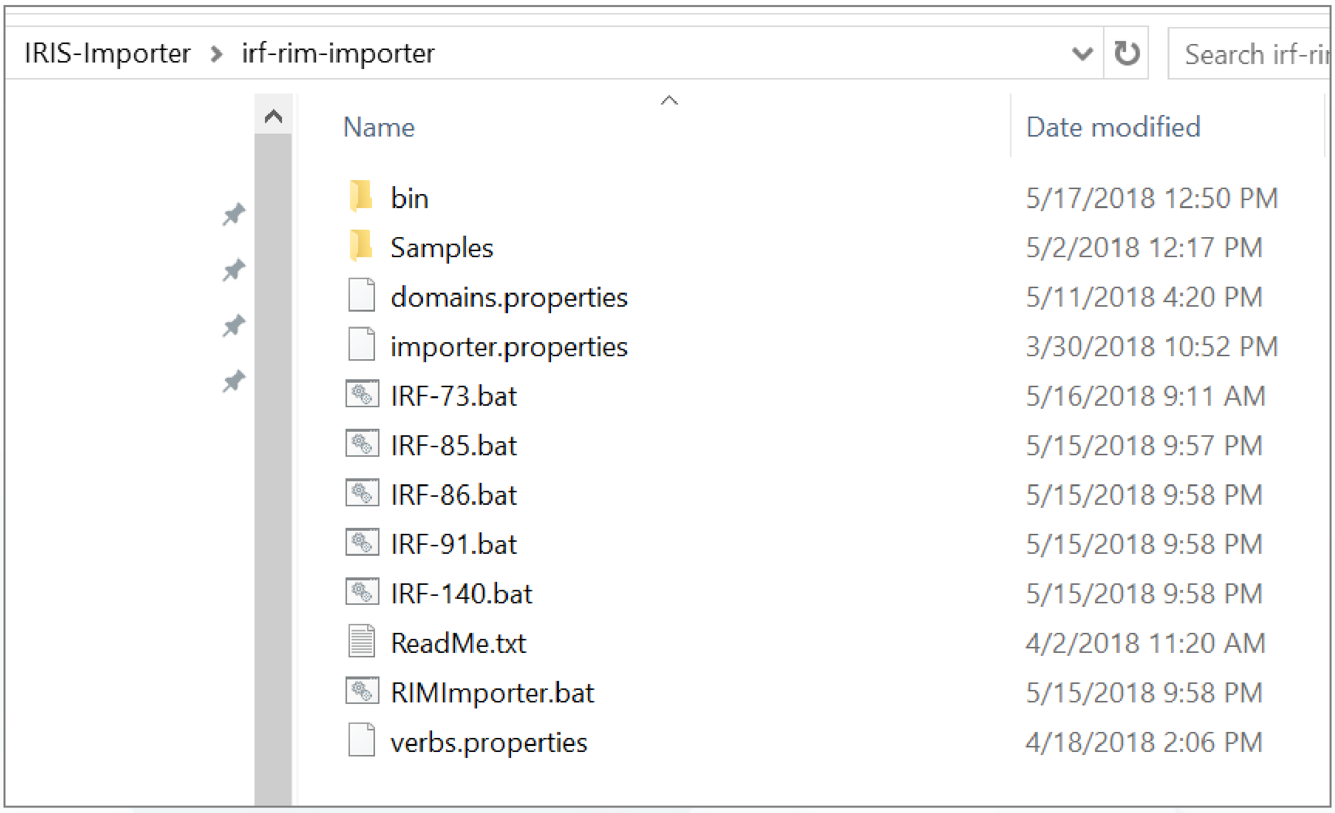
   Link : http://maven.temenosgroup.com/content/groups/temenos/com/temenos/irf/irf-rim-importer-packager/DEV.0.0-SNAPSHOT/

2. Extract the rimimporter pack.



Name	Date modified
IRIS-R18 RIM Importer Technical Specification.d...	5/2/2018 9:07 AM
TCIB	5/17/2018 12:50 PM
Nordea	5/17/2018 12:50 PM
irf-rim-importer	5/17/2018 12:50 PM
irf-config	5/14/2018 12:31 PM

3. Navigate to location **"..\irf-rim-importer"** within the extract folder.

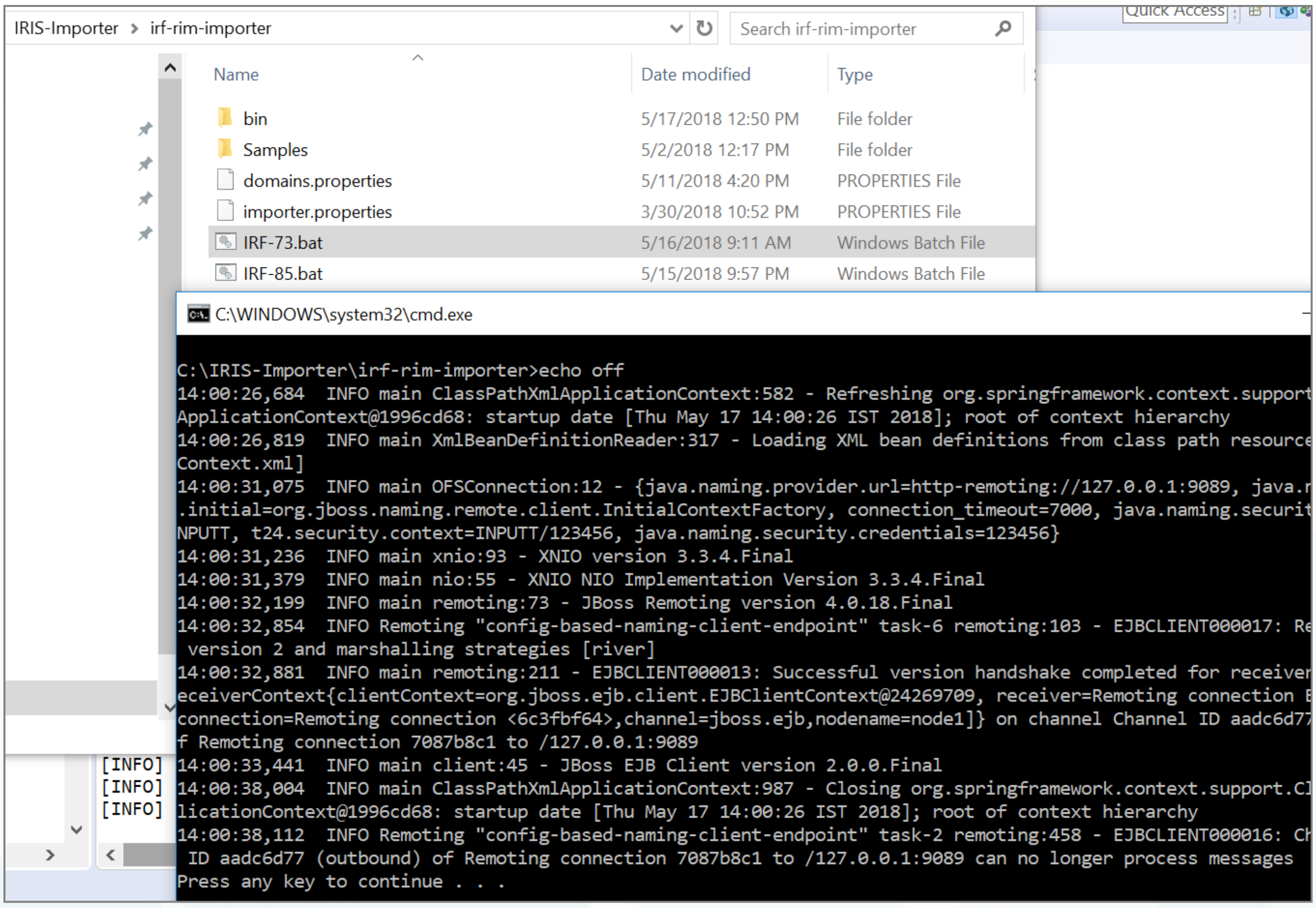


IRIS-Importer > irf-rim-importer

Search irf-ri

Name	Date modified
bin	5/17/2018 12:50 PM
Samples	5/2/2018 12:17 PM
domains.properties	5/11/2018 4:20 PM
importer.properties	3/30/2018 10:52 PM
IRF-73.bat	5/16/2018 9:11 AM
IRF-85.bat	5/15/2018 9:57 PM
IRF-86.bat	5/15/2018 9:58 PM
IRF-91.bat	5/15/2018 9:58 PM
IRF-140.bat	5/15/2018 9:58 PM
ReadMe.txt	4/2/2018 11:20 AM
RIMImporter.bat	5/15/2018 9:58 PM
verbs.properties	4/18/2018 2:06 PM

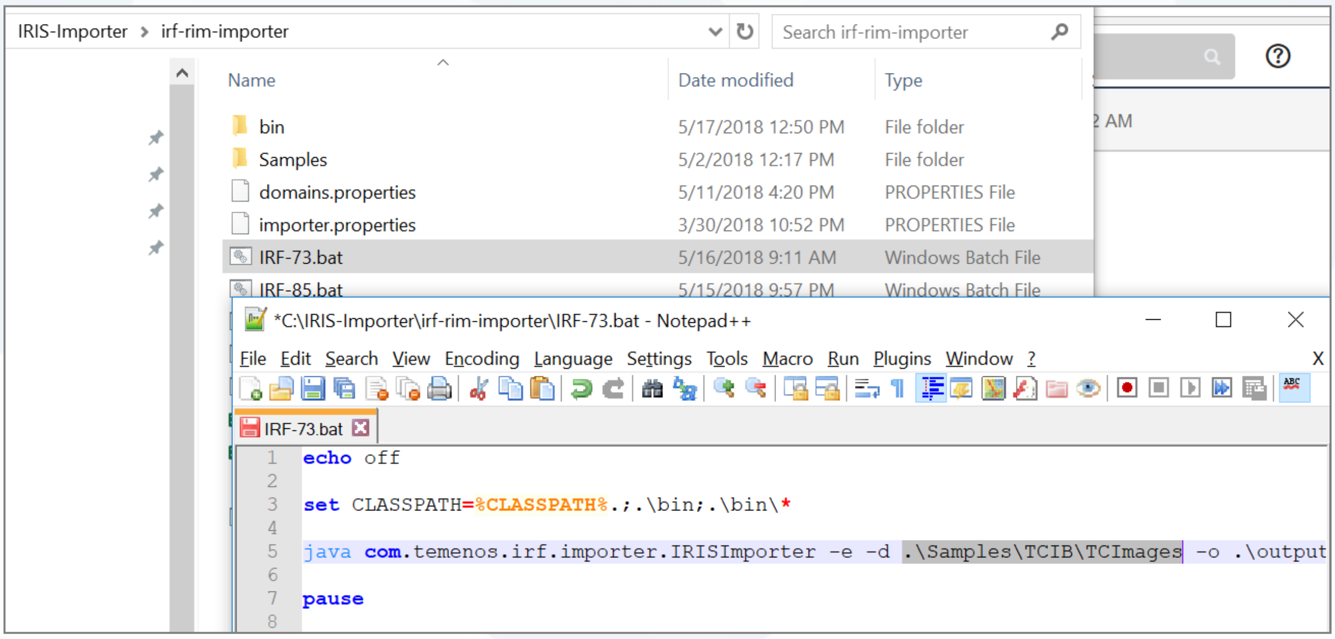
4. To Extract API's using sample RIMs (TCIB, Nordea), Run **"1.extract-apis.bat"**



Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File

```
C:\IRIS-Importer\irf-rim-importer>echo off
14:00:26,684  INFO main ClassPathXmlApplicationContext:582 - Refreshing org.springframework.context.support
ApplicationContext@1996cd68: startup date [Thu May 17 14:00:26 IST 2018]; root of context hierarchy
14:00:26,819  INFO main XmlBeanDefinitionReader:317 - Loading XML bean definitions from class path resource
Context.xml]
14:00:31,075  INFO main OFSConnection:12 - {java.naming.provider.url=http-remoting://127.0.0.1:9089, java.
.initial=org.jboss.naming.remote.client.InitialContextFactory, connection_timeout=7000, java.naming.securit
NPUTT, t24.security.context=INPUTT/123456, java.naming.security.credentials=123456}
14:00:31,236  INFO main xnio:93 - XNIO version 3.3.4.Final
14:00:31,379  INFO main nio:55 - XNIO NIO Implementation Version 3.3.4.Final
14:00:32,199  INFO main remoting:73 - JBoss Remoting version 4.0.18.Final
14:00:32,854  INFO Remoting "config-based-naming-client-endpoint" task-6 remoting:103 - EJBCLIENT000017: Re
 version 2 and marshalling strategies [river]
14:00:32,881  INFO main remoting:211 - EJBCLIENT000013: Successful version handshake completed for receiver
eceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@24269709, receiver=Remoting connection E
connection=Remoting connection <6c3fbf64>,channel=jboss.ejb,nodename=node1]} on channel Channel ID aadc6d77
f Remoting connection 7087b8c1 to /127.0.0.1:9089
14:00:33,441  INFO main client:45 - JBoss EJB Client version 2.0.0.Final
14:00:38,004  INFO main ClassPathXmlApplicationContext:987 - Closing org.springframework.context.support.Cl
licationContext@1996cd68: startup date [Thu May 17 14:00:26 IST 2018]; root of context hierarchy
14:00:38,112  INFO Remoting "config-based-naming-client-endpoint" task-2 remoting:458 - EJBCLIENT000016: Ch
 ID aadc6d77 (outbound) of Remoting connection 7087b8c1 to /127.0.0.1:9089 can no longer process messages
Press any key to continue . . .
```

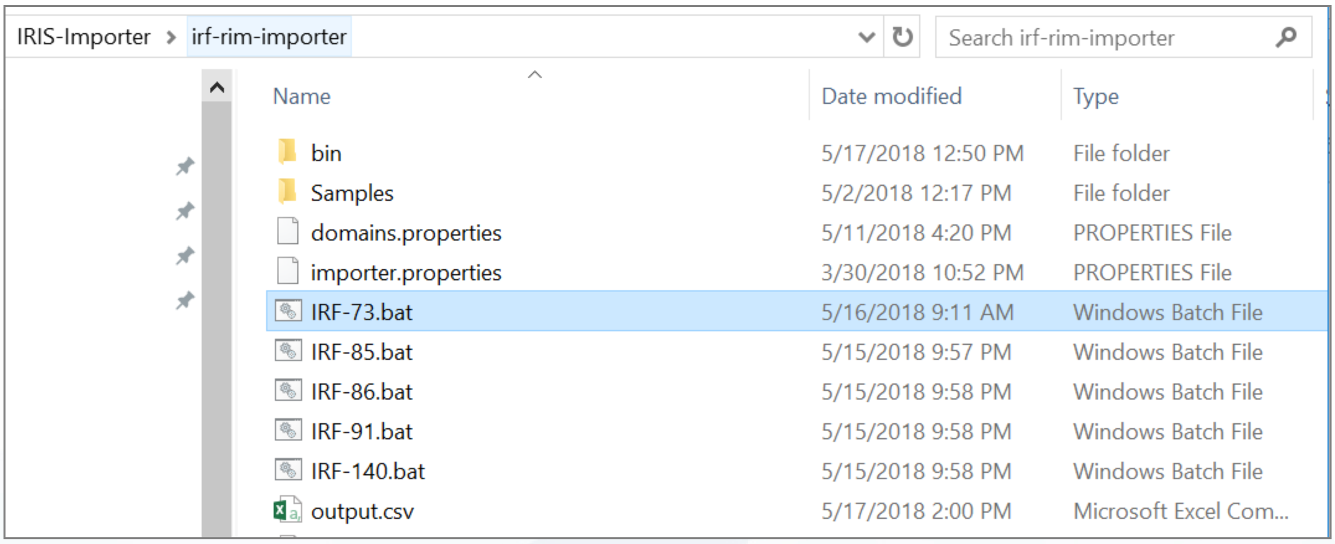
Note: Open Run IRF-73.bat and Edit the rim Source Path


IRIS-Importer > irf-rim-importer
Search irf-rim-importer
*C:\IRIS-Importer\irf-rim-importer\IRF-73.bat - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
IRF-73.bat

Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File

```
echo off

set CLASSPATH=%CLASSPATH%.;.\bin;.\bin\*

java com.temenos.irf.importer.IRISImporter -e -d .\Samples\TCIB\TCImages -o .\output

pause

```

5. Extracted APIs will be written and found in file **"output.csv"**

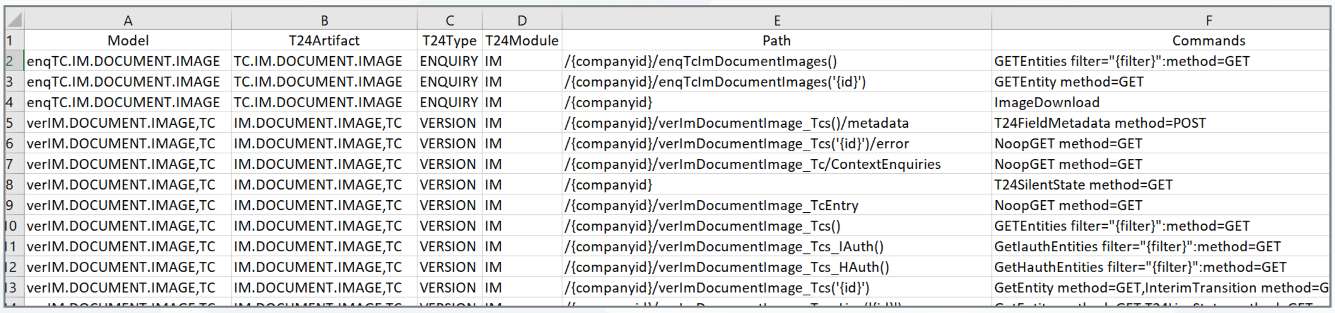


IRIS-Importer > irf-rim-importer

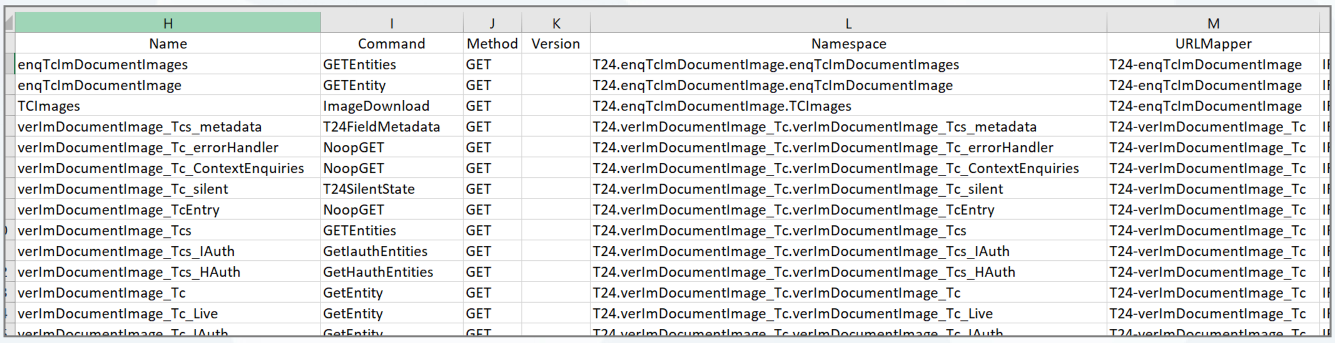
Search irf-rim-importer

Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File
IRF-86.bat	5/15/2018 9:58 PM	Windows Batch File
IRF-91.bat	5/15/2018 9:58 PM	Windows Batch File
IRF-140.bat	5/15/2018 9:58 PM	Windows Batch File
output.csv	5/17/2018 2:00 PM	Microsoft Excel Com...

6. Verify and Confirm the API extracted from RIMS and edit, if required

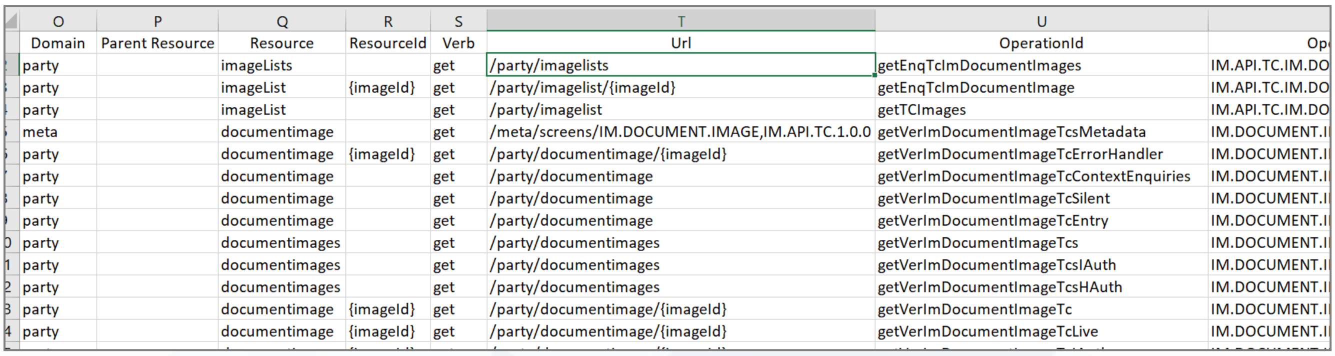


	A	B	C	D	E	F
1	Model	T24Artifact	T24Type	T24Module	Path	Commands
2	enqTC.IM.DOCUMENT.IMAGE	TC.IM.DOCUMENT.IMAGE	ENQUIRY	IM	/{companyid}/enqTcImDocumentImages()	GETEntities filter="{filter}":method=GET
3	enqTC.IM.DOCUMENT.IMAGE	TC.IM.DOCUMENT.IMAGE	ENQUIRY	IM	/{companyid}/enqTcImDocumentImages('{id}')	GETEntity method=GET
4	enqTC.IM.DOCUMENT.IMAGE	TC.IM.DOCUMENT.IMAGE	ENQUIRY	IM	/{companyid}	ImageDownload
5	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tcs()/metadata	T24FieldMetadata method=POST
6	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tcs('{id}')/error	NoopGET method=GET
7	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tc/ContextEnquiries	NoopGET method=GET
8	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}	T24SilentState method=GET
9	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_TcEntry	NoopGET method=GET
10	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tcs()	GETEntities filter="{filter}":method=GET
11	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tcs_IAuth()	GetIauthEntities filter="{filter}":method=GET
12	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tcs_HAuth()	GetHauthEntities filter="{filter}":method=GET
13	verIM.DOCUMENT.IMAGE,TC	IM.DOCUMENT.IMAGE,TC	VERSION	IM	/{companyid}/verImDocumentImage_Tcs('{id}')	GetEntity method=GET,InterimTransition method=G

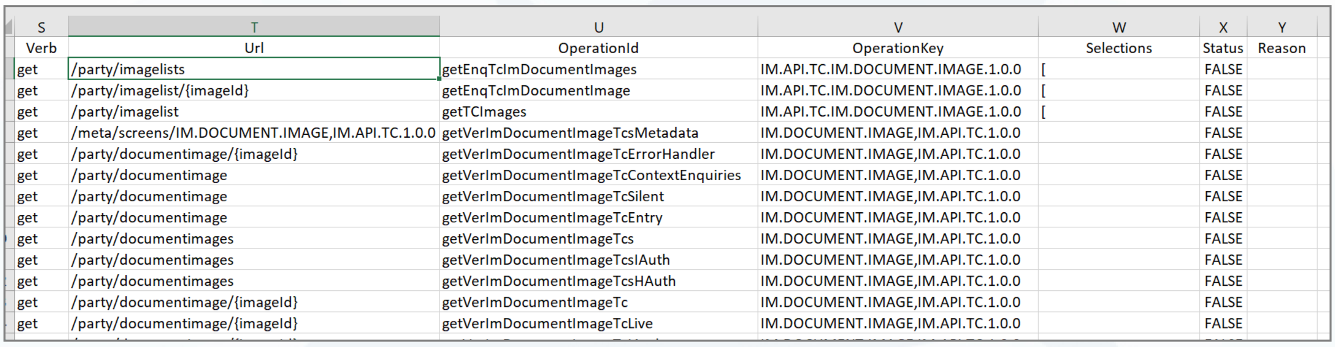


H	I	J	K	L	M
Name	Command	Method	Version	Namespace	URLMapper
enqTcImDocumentImages	GETEntities	GET		T24.enqTcImDocumentImage.enqTcImDocumentImages	T24-enqTcImDocumentImage
enqTcImDocumentImage	GETEntity	GET		T24.enqTcImDocumentImage.enqTcImDocumentImage	T24-enqTcImDocumentImage
TCImages	ImageDownload	GET		T24.enqTcImDocumentImage.TCImages	T24-enqTcImDocumentImage
verImDocumentImage_Tcs_metadata	T24FieldMetadata	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tcs_metadata	T24-verImDocumentImage_Tc
verImDocumentImage_Tc_errorHandler	NoopGET	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tc_errorHandler	T24-verImDocumentImage_Tc
verImDocumentImage_Tc_ContextEnquiries	NoopGET	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tc_ContextEnquiries	T24-verImDocumentImage_Tc
verImDocumentImage_Tc_silent	T24SilentState	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tc_silent	T24-verImDocumentImage_Tc
verImDocumentImage_TcEntry	NoopGET	GET		T24.verImDocumentImage_Tc.verImDocumentImage_TcEntry	T24-verImDocumentImage_Tc
verImDocumentImage_Tcs	GETEntities	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tcs	T24-verImDocumentImage_Tc
verImDocumentImage_Tcs_IAuth	GetIauthEntities	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tcs_IAuth	T24-verImDocumentImage_Tc
verImDocumentImage_Tcs_HAuth	GetHauthEntities	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tcs_HAuth	T24-verImDocumentImage_Tc
verImDocumentImage_Tc	GetEntity	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tc	T24-verImDocumentImage_Tc
verImDocumentImage_Tc_Live	GetEntity	GET		T24.verImDocumentImage_Tc.verImDocumentImage_Tc_Live	T24-verImDocumentImage_Tc

Note: Editable Columns Domain, Parent Resource, Resource, ResourceId, Verb, OperationId, OperationKey, Selections

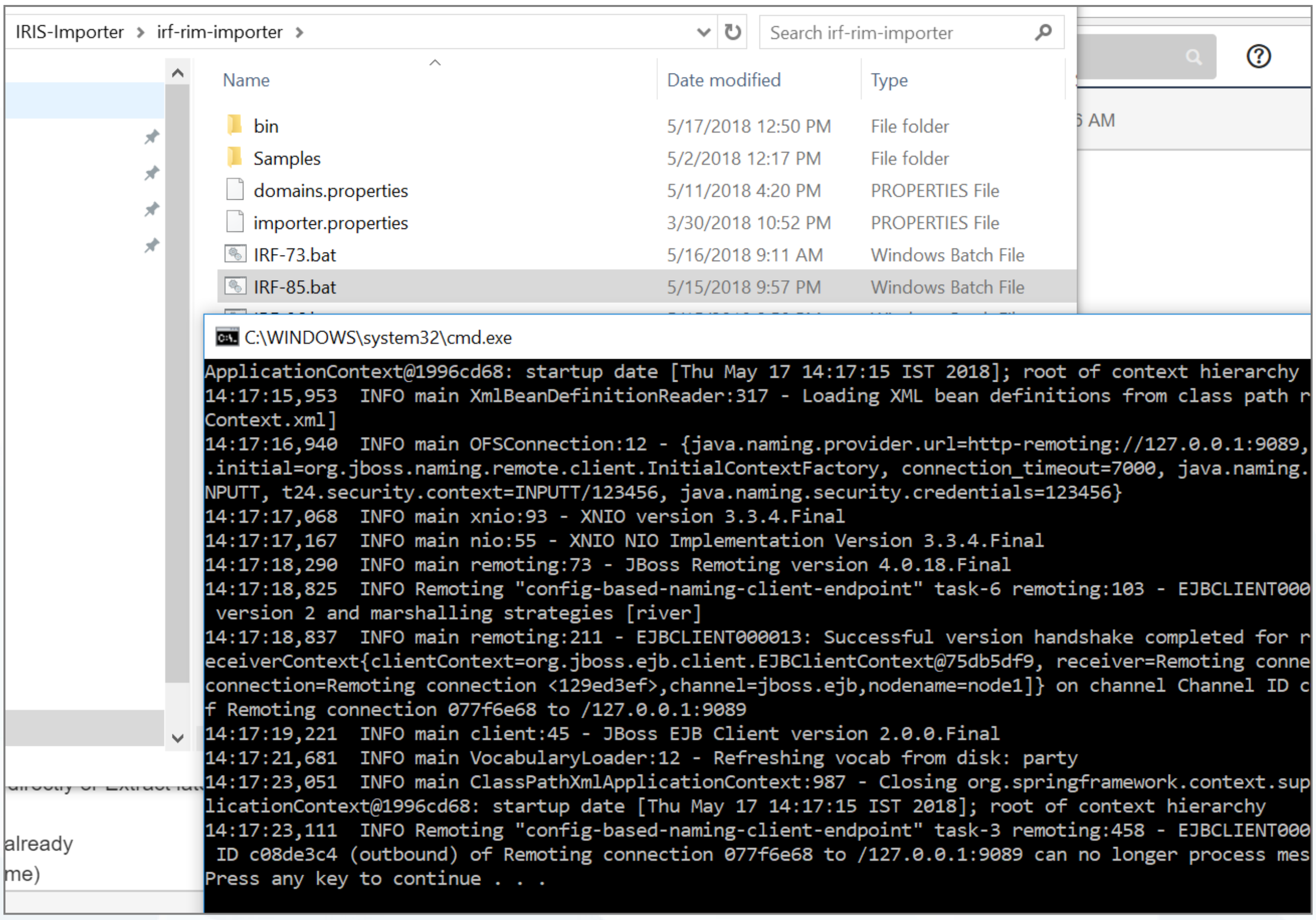


O	P	Q	R	S	T	U	
Domain	Parent Resource	Resource	ResourceId	Verb	Url	OperationId	Op
party		imageLists		get	/party/imagelists	getEnqTcImDocumentImages	IM.API.TC.IM.DO
party		imageList	{imageId}	get	/party/imagelist/{imageId}	getEnqTcImDocumentImage	IM.API.TC.IM.DO
party		imageList		get	/party/imagelist	getTCImages	IM.API.TC.IM.DO
meta		documentimage		get	/meta/screens/IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0	getVerImDocumentImageTcsMetadata	IM.DOCUMENT.I
party		documentimage	{imageId}	get	/party/documentimage/{imageId}	getVerImDocumentImageTcErrorHandler	IM.DOCUMENT.I
party		documentimage		get	/party/documentimage	getVerImDocumentImageTcContextEnquiries	IM.DOCUMENT.I
party		documentimage		get	/party/documentimage	getVerImDocumentImageTcSilent	IM.DOCUMENT.I
party		documentimage		get	/party/documentimage	getVerImDocumentImageTcEntry	IM.DOCUMENT.I
party		documentimages		get	/party/documentimages	getVerImDocumentImageTcs	IM.DOCUMENT.I
party		documentimages		get	/party/documentimages	getVerImDocumentImageTcsIAuth	IM.DOCUMENT.I
party		documentimages		get	/party/documentimages	getVerImDocumentImageTcsHAuth	IM.DOCUMENT.I
party		documentimage	{imageId}	get	/party/documentimage/{imageId}	getVerImDocumentImageTc	IM.DOCUMENT.I
party		documentimage	{imageId}	get	/party/documentimage/{imageId}	getVerImDocumentImageTcLive	IM.DOCUMENT.I



S	T	U	V	W	X	Y
Verb	Url	OperationId	OperationKey	Selections	Status	Reason
get	/party/imagelists	getEnqTcImDocumentImages	IM.API.TC.IM.DOCUMENT.IMAGE.1.0.0	[	FALSE	
get	/party/imagelist/{imageId}	getEnqTcImDocumentImage	IM.API.TC.IM.DOCUMENT.IMAGE.1.0.0	[	FALSE	
get	/party/imagelist	getTCImages	IM.API.TC.IM.DOCUMENT.IMAGE.1.0.0	[	FALSE	
get	/meta/screens/IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0	getVerImDocumentImageTcsMetadata	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimage/{imageId}	getVerImDocumentImageTcErrorHandler	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimage	getVerImDocumentImageTcContextEnquiries	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimage	getVerImDocumentImageTcSilent	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimage	getVerImDocumentImageTcEntry	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimages	getVerImDocumentImageTcs	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimages	getVerImDocumentImageTcsIAuth	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimages	getVerImDocumentImageTcsHAuth	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimage/{imageId}	getVerImDocumentImageTc	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	
get	/party/documentimage/{imageId}	getVerImDocumentImageTcLive	IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0		FALSE	

7. Validate the extracted API's against the IRISX conventions for conformance, Run **"2.validate-apis.bat"**


IRIS-Importer > irf-rim-importer >

Search irf-rim-importer

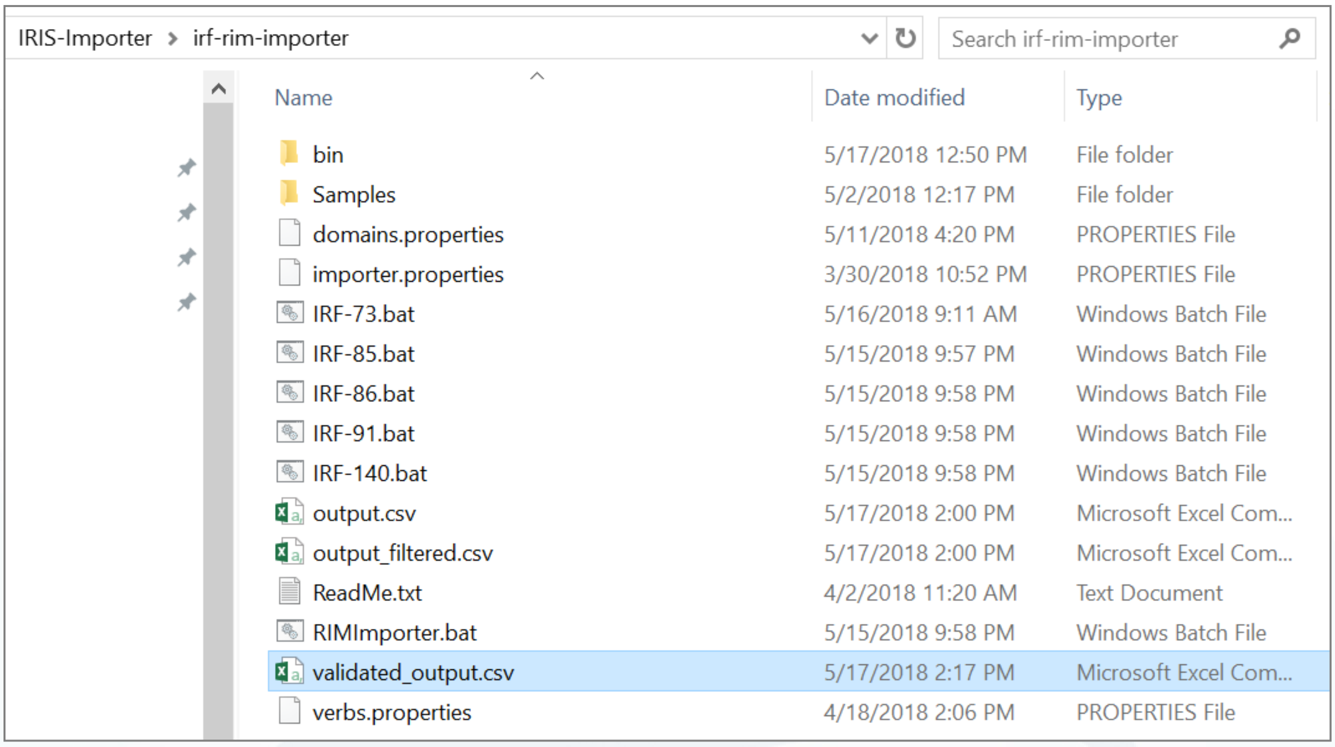
Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File

already
me)

C:\WINDOWS\system32\cmd.exe

```
ApplicationContext@1996cd68: startup date [Thu May 17 14:17:15 IST 2018]; root of context hierarchy
14:17:15,953  INFO main XmlBeanDefinitionReader:317 - Loading XML bean definitions from class path r
Context.xml]
14:17:16,940  INFO main OFSConnection:12 - {java.naming.provider.url=http-remoting://127.0.0.1:9089,
.initial=org.jboss.naming.remote.client.InitialContextFactory, connection_timeout=7000, java.naming.
NPUTT, t24.security.context=INPUTT/123456, java.naming.security.credentials=123456}
14:17:17,068  INFO main xnio:93 - XNIO version 3.3.4.Final
14:17:17,167  INFO main nio:55 - XNIO NIO Implementation Version 3.3.4.Final
14:17:18,290  INFO main remoting:73 - JBoss Remoting version 4.0.18.Final
14:17:18,825  INFO Remoting "config-based-naming-client-endpoint" task-6 remoting:103 - EJBCLIENT000
 version 2 and marshalling strategies [river]
14:17:18,837  INFO main remoting:211 - EJBCLIENT000013: Successful version handshake completed for r
eceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@75db5df9, receiver=Remoting conne
connection=Remoting connection <129ed3ef>,channel=jboss.ejb,nodename=node1]} on channel Channel ID c
f Remoting connection 077f6e68 to /127.0.0.1:9089
14:17:19,221  INFO main client:45 - JBoss EJB Client version 2.0.0.Final
14:17:21,681  INFO main VocabularyLoader:12 - Refreshing vocab from disk: party
14:17:23,051  INFO main ClassPathXmlApplicationContext:987 - Closing org.springframework.context.sup
licationContext@1996cd68: startup date [Thu May 17 14:17:15 IST 2018]; root of context hierarchy
14:17:23,111  INFO Remoting "config-based-naming-client-endpoint" task-3 remoting:458 - EJBCLIENT000
 ID c08de3c4 (outbound) of Remoting connection 077f6e68 to /127.0.0.1:9089 can no longer process mes
Press any key to continue . . .
```

8. The validation status along with the reason will be updated in file **"validated_output.csv"**

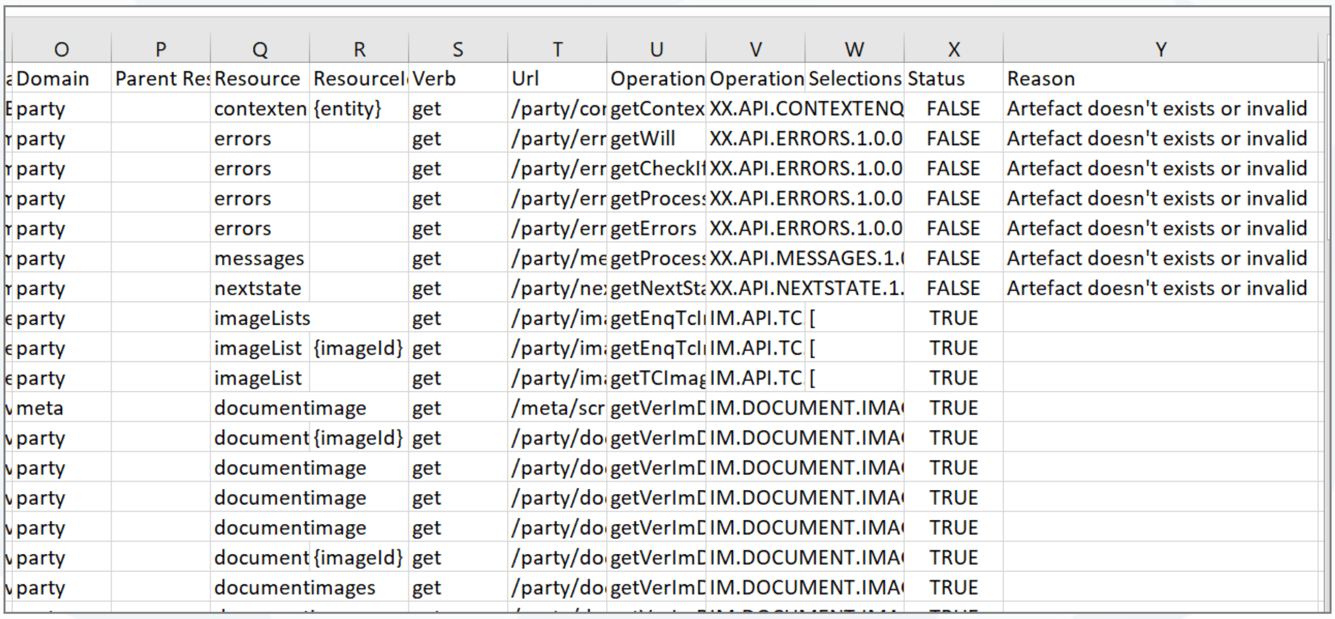


IRIS-Importer > irf-rim-importer

Search irf-rim-importer

Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File
IRF-86.bat	5/15/2018 9:58 PM	Windows Batch File
IRF-91.bat	5/15/2018 9:58 PM	Windows Batch File
IRF-140.bat	5/15/2018 9:58 PM	Windows Batch File
output.csv	5/17/2018 2:00 PM	Microsoft Excel Com...
output_filtered.csv	5/17/2018 2:00 PM	Microsoft Excel Com...
ReadMe.txt	4/2/2018 11:20 AM	Text Document
RIMImporter.bat	5/15/2018 9:58 PM	Windows Batch File
validated_output.csv	5/17/2018 2:17 PM	Microsoft Excel Com...
verbs.properties	4/18/2018 2:06 PM	PROPERTIES File

9. Refer Status column for conformance status and refer Reason column for the failure cause.

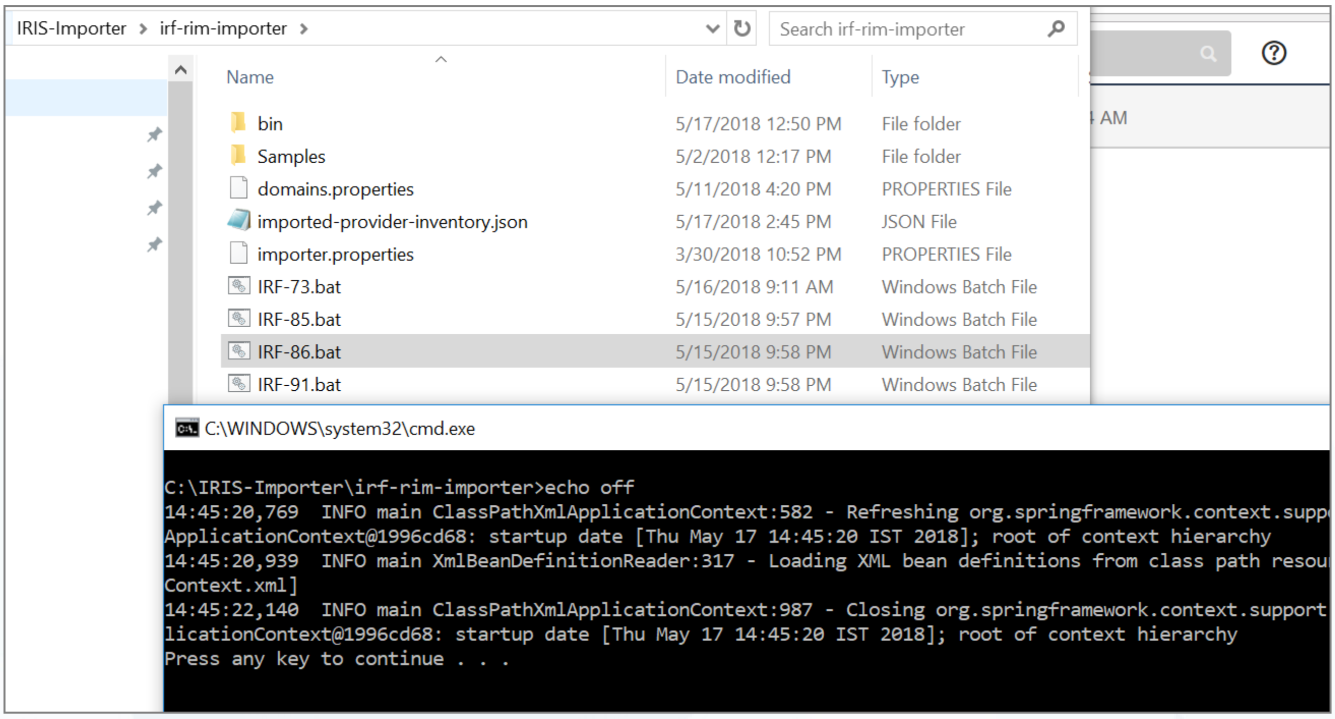


O	P	Q	R	S	T	U	V	W	X	Y
Domain	Parent Res	Resource	ResourceI	Verb	Url	Operation	Operation	Selections	Status	Reason
party		contexten	{entity}	get	/party/co	getContex	XX.API.CONTEXTENQ		FALSE	Artefact doesn't exists or invalid
party		errors		get	/party/err	getWill	XX.API.ERRORS.1.0.0		FALSE	Artefact doesn't exists or invalid
party		errors		get	/party/err	getCheckI	XX.API.ERRORS.1.0.0		FALSE	Artefact doesn't exists or invalid
party		errors		get	/party/err	getProces	XX.API.ERRORS.1.0.0		FALSE	Artefact doesn't exists or invalid
party		errors		get	/party/err	getErrors	XX.API.ERRORS.1.0.0		FALSE	Artefact doesn't exists or invalid
party		messages		get	/party/me	getProces	XX.API.MESSAGES.1.		FALSE	Artefact doesn't exists or invalid
party		nextstate		get	/party/ne	getNextSt	XX.API.NEXTSTATE.1.		FALSE	Artefact doesn't exists or invalid
party		imageLists		get	/party/im	getEnqTcl	IM.API.TC	[	TRUE	
party		imageList	{imageId}	get	/party/im	getEnqTcl	IM.API.TC	[	TRUE	
party		imageList		get	/party/im	getTCImag	IM.API.TC	[	TRUE	
meta		documentimage		get	/meta/scr	getVerImD	IM.DOCUMENT.IMA		TRUE	
party		document	{imageId}	get	/party/do	getVerImD	IM.DOCUMENT.IMA		TRUE	
party		documentimage		get	/party/do	getVerImD	IM.DOCUMENT.IMA		TRUE	
party		documentimage		get	/party/do	getVerImD	IM.DOCUMENT.IMA		TRUE	
party		documentimage		get	/party/do	getVerImD	IM.DOCUMENT.IMA		TRUE	
party		document	{imageId}	get	/party/do	getVerImD	IM.DOCUMENT.IMA		TRUE	
party		documentimages		get	/party/do	getVerImD	IM.DOCUMENT.IMA		TRUE	

10. Correct and re-run steps 7-9 until Status columns gets TRUE.

Note: Update vocabulary or artifact if resource or property doesn't exist (VERSION, ENQUIRY)

11. The extracted API's with **Status = TRUE** are ready for the inventory generation, Run **"3.generate-inventory.bat"**



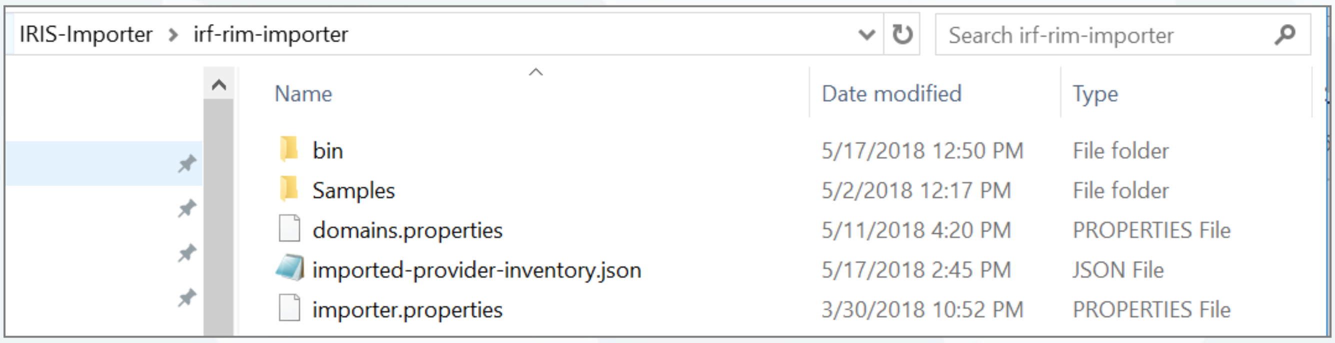
IRIS-Importer > irf-rim-importer >

Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
imported-provider-inventory.json	5/17/2018 2:45 PM	JSON File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File
IRF-86.bat	5/15/2018 9:58 PM	Windows Batch File
IRF-91.bat	5/15/2018 9:58 PM	Windows Batch File

C:\WINDOWS\system32\cmd.exe

```
C:\IRIS-Importer\irf-rim-importer>echo off
14:45:20,769  INFO main ClassPathXmlApplicationContext:582 - Refreshing org.springframework.context.supp
ApplicationContext@1996cd68: startup date [Thu May 17 14:45:20 IST 2018]; root of context hierarchy
14:45:20,939  INFO main XmlBeanDefinitionReader:317 - Loading XML bean definitions from class path resou
Context.xml]
14:45:22,140  INFO main ClassPathXmlApplicationContext:987 - Closing org.springframework.context.support
licationContext@1996cd68: startup date [Thu May 17 14:45:20 IST 2018]; root of context hierarchy
Press any key to continue . . .
```

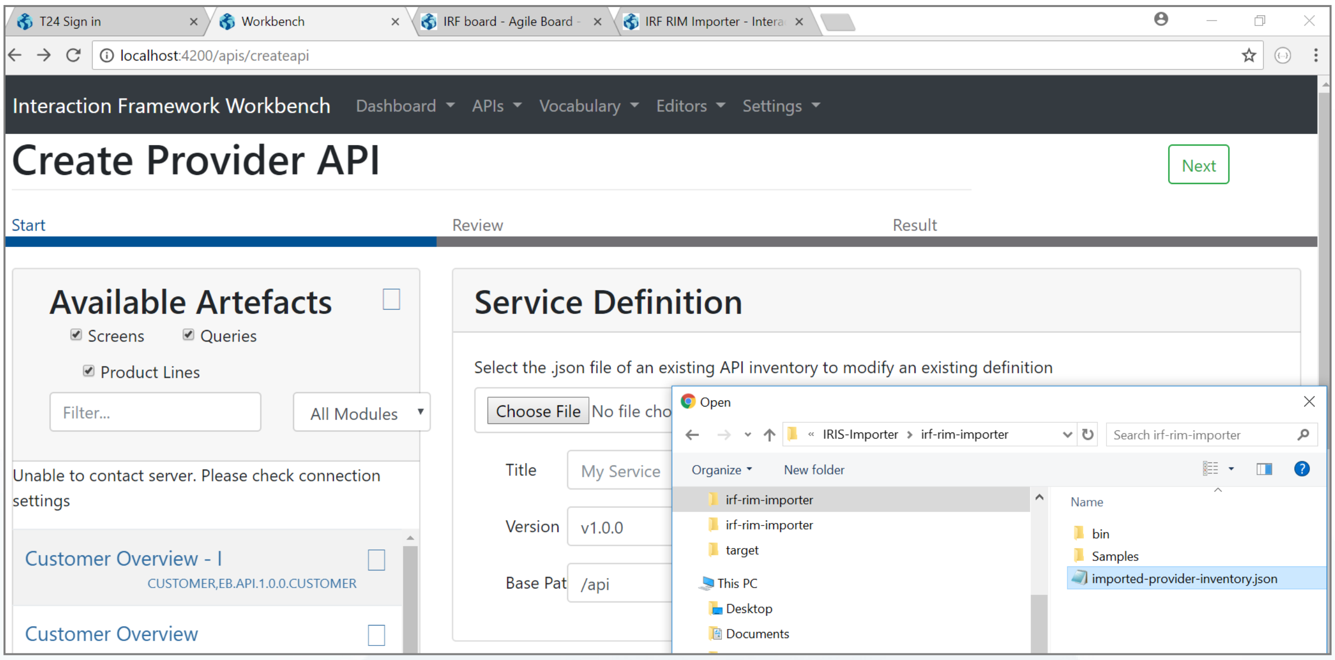
12. The inventory file **"imported-provider-inventory.json"** will be generated in IRISX service generation schema.



IRIS-Importer > irf-rim-importer

Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
imported-provider-inventory.json	5/17/2018 2:45 PM	JSON File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File

Note: Verify the correctness of import via **Workbench** & to generate the service using **Workbench**



Create Provider API

Available Artefacts

☑ Screens ☑ Queries

☑ Product Lines

Filter...

All Modules

Unable to contact server. Please check connection settings

Customer Overview - I
CUSTOMER,EB.API.1.0.0.CUSTOMER

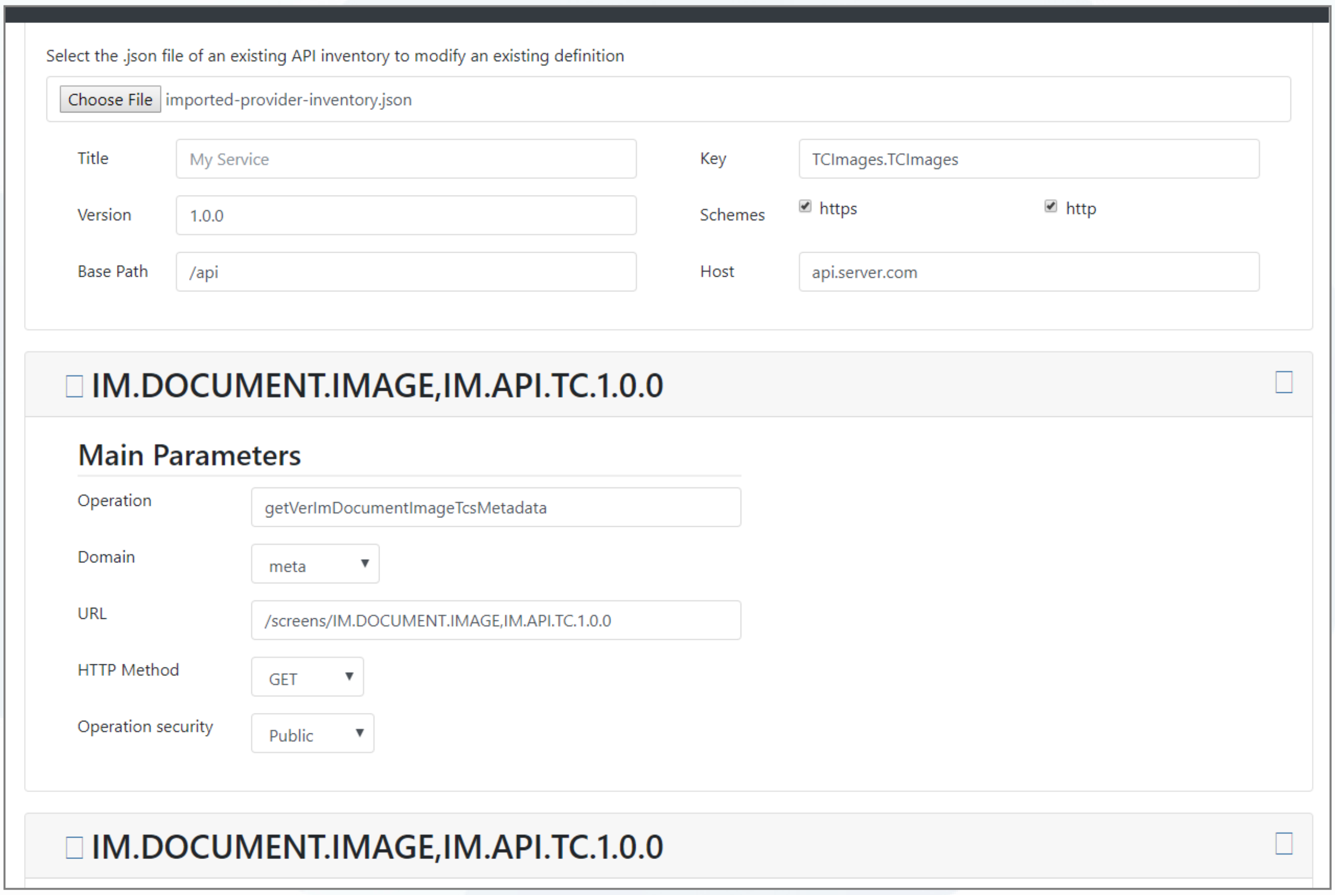
Customer Overview

Service Definition

Select the .json file of an existing API inventory to modify an existing definition

Choose File No file cho

Title	My Service
Version	v1.0.0
Base Pat	/api



Select the .json file of an existing API inventory to modify an existing definition

Choose File imported-provider-inventory.json

Title	My Service	Key	TCImages.TCImages
Version	1.0.0	Schemes	☑ https ☑ http
Base Path	/api	Host	api.server.com

IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0

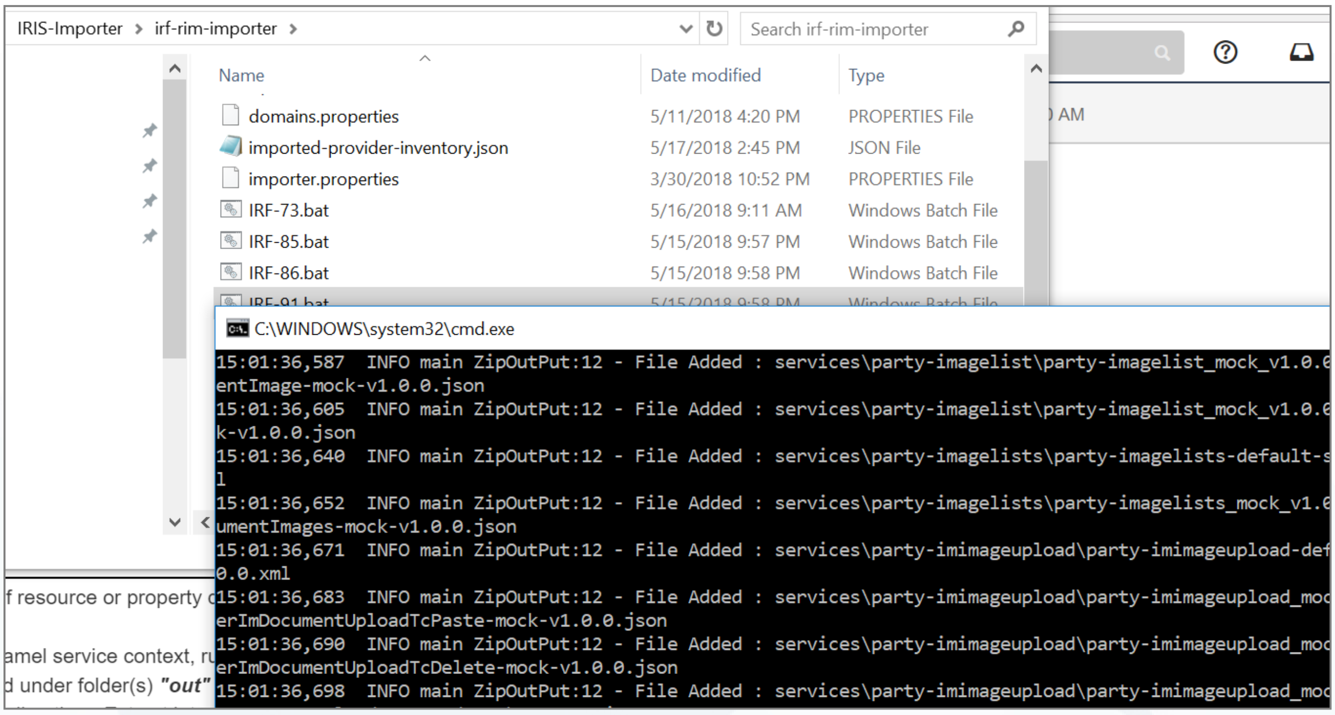
Main Parameters

Operation	getVerImDocumentImageTcsMetadata
Domain	meta
URL	/screens/IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0
HTTP Method	GET
Operation security	Public

IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0

Note: Update vocabulary or artifact if resource or property doesn't exist (VERSION, ENQUIRY)

13. The inventory is ready to generate camel service context, run **"4.generate-service.bat"**



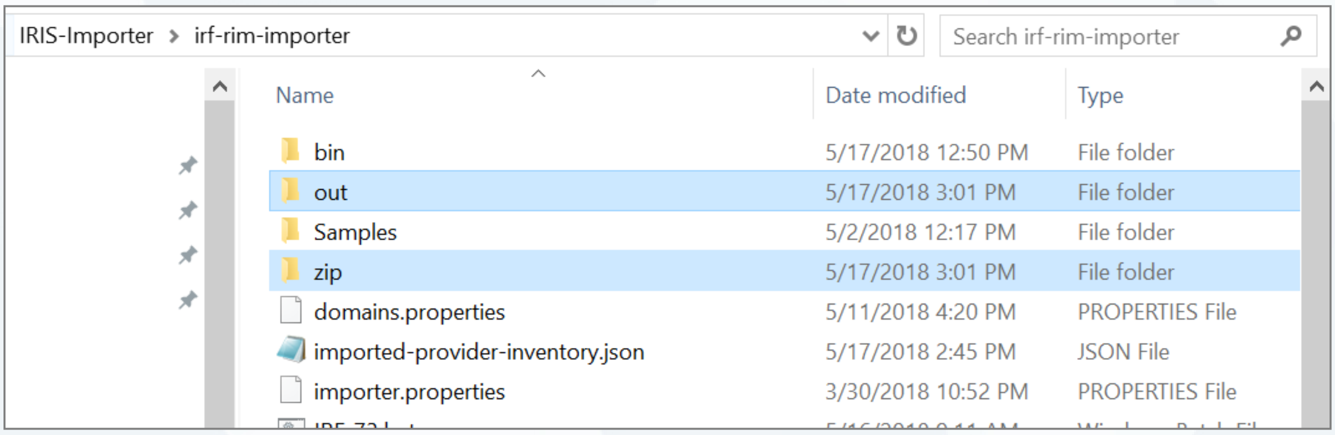
IRIS-Importer > irf-rim-importer >

Name	Date modified	Type
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
imported-provider-inventory.json	5/17/2018 2:45 PM	JSON File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File
IRF-73.bat	5/16/2018 9:11 AM	Windows Batch File
IRF-85.bat	5/15/2018 9:57 PM	Windows Batch File
IRF-86.bat	5/15/2018 9:58 PM	Windows Batch File
IRF-91.bat	5/15/2018 9:58 PM	Windows Batch File

C:\WINDOWS\system32\cmd.exe

```
15:01:36,587  INFO main ZipOutPut:12 - File Added : services\party-imagelist\party-imagelist_mock_v1.0.0
entImage-mock-v1.0.0.json
15:01:36,605  INFO main ZipOutPut:12 - File Added : services\party-imagelist\party-imagelist_mock_v1.0.0
k-v1.0.0.json
15:01:36,640  INFO main ZipOutPut:12 - File Added : services\party-imagelists\party-imagelists-default-s
l
15:01:36,652  INFO main ZipOutPut:12 - File Added : services\party-imagelists\party-imagelists_mock_v1.0
umentImages-mock-v1.0.0.json
15:01:36,671  INFO main ZipOutPut:12 - File Added : services\party-imimageupload\party-imimageupload-def
0.0.xml
15:01:36,683  INFO main ZipOutPut:12 - File Added : services\party-imimageupload\party-imimageupload_moc
erImDocumentUploadTcPaste-mock-v1.0.0.json
15:01:36,690  INFO main ZipOutPut:12 - File Added : services\party-imimageupload\party-imimageupload_moc
erImDocumentUploadTcDelete-mock-v1.0.0.json
15:01:36,698  INFO main ZipOutPut:12 - File Added : services\party-imimageupload\party-imimageupload_moc
```

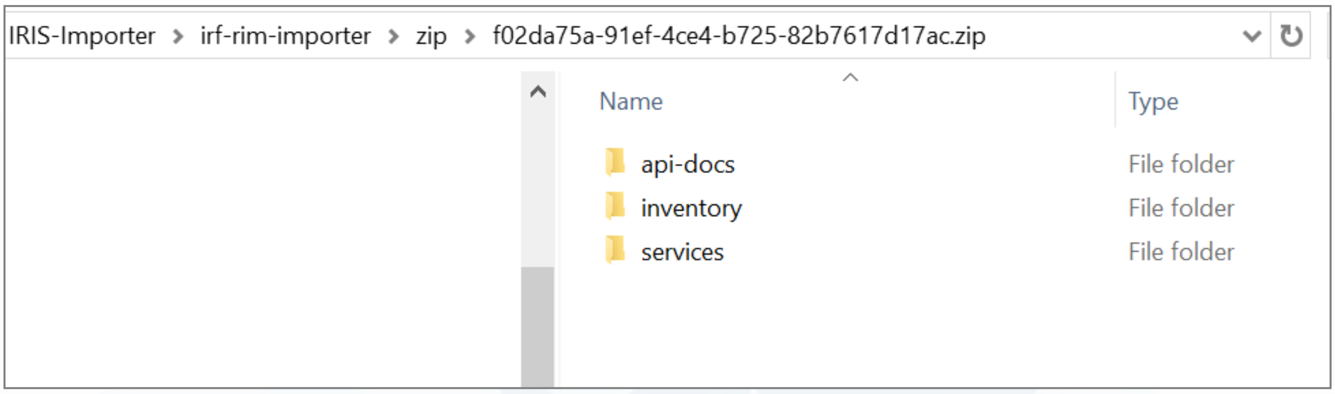
14. The service context will be generated under folder(s) **"out"** and **"zip"** respectively.



IRIS-Importer > irf-rim-importer

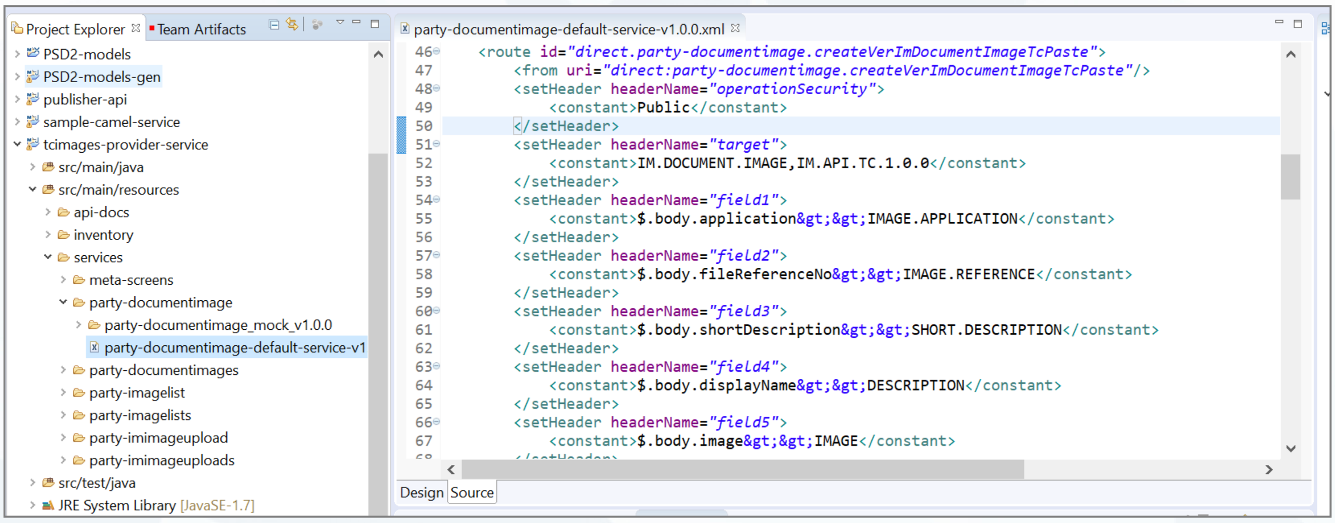
Name	Date modified	Type
bin	5/17/2018 12:50 PM	File folder
out	5/17/2018 3:01 PM	File folder
Samples	5/2/2018 12:17 PM	File folder
zip	5/17/2018 3:01 PM	File folder
domains.properties	5/11/2018 4:20 PM	PROPERTIES File
imported-provider-inventory.json	5/17/2018 2:45 PM	JSON File
importer.properties	3/30/2018 10:52 PM	PROPERTIES File

15. Copy the service context from **"out"** directly or Extract latest archive from **"zip"** to IRF Service Project **"src/main/resources"**



IRIS-Importer > irf-rim-importer > zip > f02da75a-91ef-4ce4-b725-82b7617d17ac.zip

Name	Type
api-docs	File folder
inventory	File folder
services	File folder



```
    <route id="direct.party-documentimage.createVerImDocumentImageTcPaste">
        <from uri="direct:party-documentimage.createVerImDocumentImageTcPaste"/>
        <setHeader headerName="operationSecurity">
            <constant>Public</constant>
        </setHeader>
        <setHeader headerName="target">
            <constant>IM.DOCUMENT.IMAGE,IM.API.TC.1.0.0</constant>
        </setHeader>
        <setHeader headerName="field1">
            <constant>$.body.application&gt;&gt;IMAGE.APPLICATION</constant>
        </setHeader>
        <setHeader headerName="field2">
            <constant>$.body.fileReferenceNo&gt;&gt;IMAGE.REFERENCE</constant>
        </setHeader>
        <setHeader headerName="field3">
            <constant>$.body.shortDescription&gt;&gt;SHORT.DESCRIPTION</constant>
        </setHeader>
        <setHeader headerName="field4">
            <constant>$.body.displayName&gt;&gt;DESCRIPTION</constant>
        </setHeader>
        <setHeader headerName="field5">
            <constant>$.body.image&gt;&gt;IMAGE</constant>
```

16. Build service project.

    Note: Refer documentation on how to create and build IRF Service Project

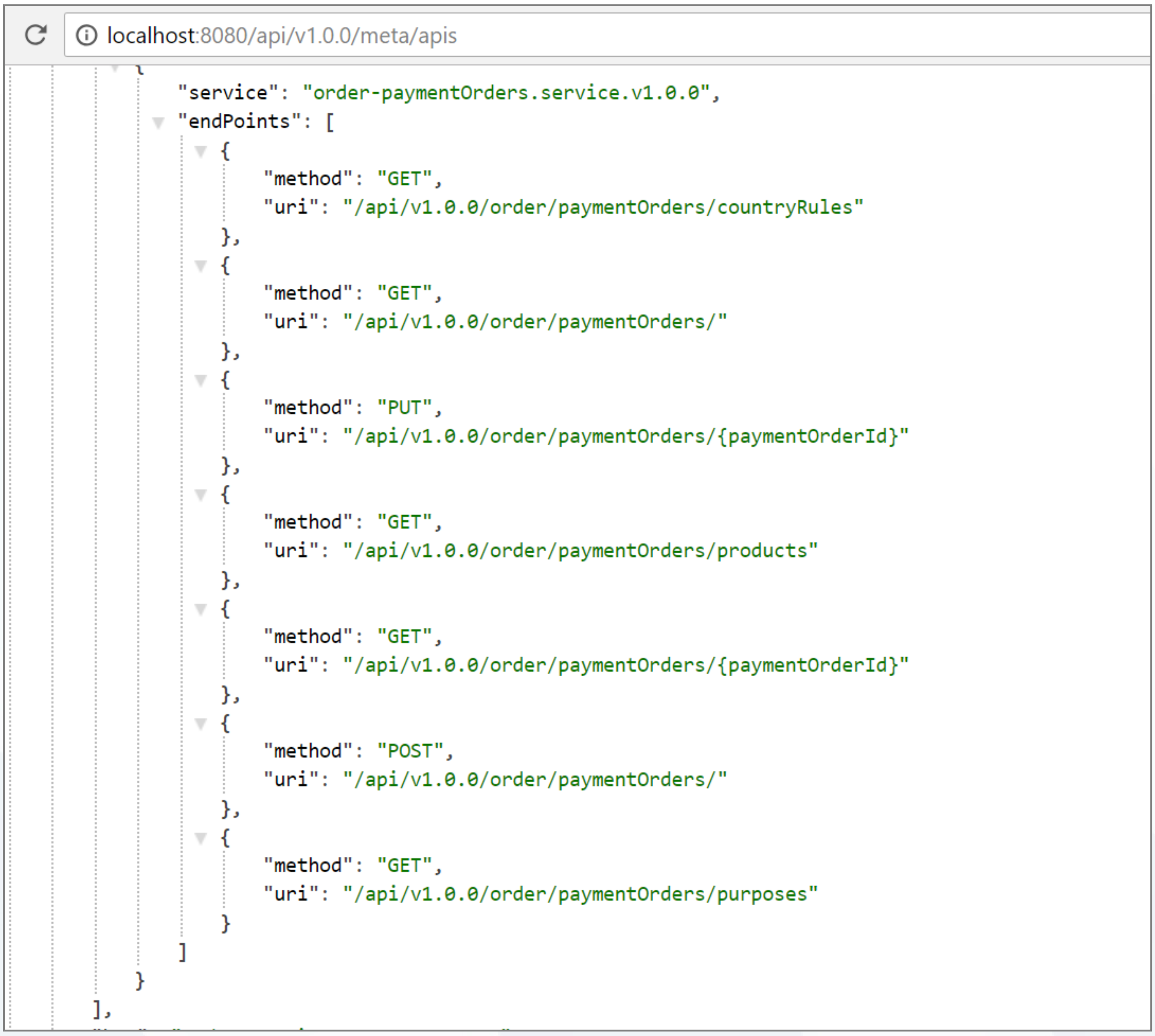
17. Run container project, if not running already.

    Note: Refer documentation on how to create, build and run IRF Service Container Project

18. Open REST Client or Browser (Chrome)

    Note: In case, you choose Browser then appropriate extensions are installed to view JSON e.g JSON Formatter

19. Enter http://localhost:8080/api/v1.0.0/meta/apis"



```
        "service": "order-paymentOrders.service.v1.0.0",
        "endPoints": [
            {
                "method": "GET",
                "uri": "/api/v1.0.0/order/paymentOrders/countryRules"
            },
            {
                "method": "GET",
                "uri": "/api/v1.0.0/order/paymentOrders/"
            },
            {
                "method": "PUT",
                "uri": "/api/v1.0.0/order/paymentOrders/{paymentOrderId}"
            },
            {
                "method": "GET",
                "uri": "/api/v1.0.0/order/paymentOrders/products"
            },
            {
                "method": "GET",
                "uri": "/api/v1.0.0/order/paymentOrders/{paymentOrderId}"
            },
            {
                "method": "POST",
                "uri": "/api/v1.0.0/order/paymentOrders/"
            },
            {
                "method": "GET",
                "uri": "/api/v1.0.0/order/paymentOrders/purposes"
            }
        ]
    }
],
```

20. Find the created service listed with all imported endpoints listed.

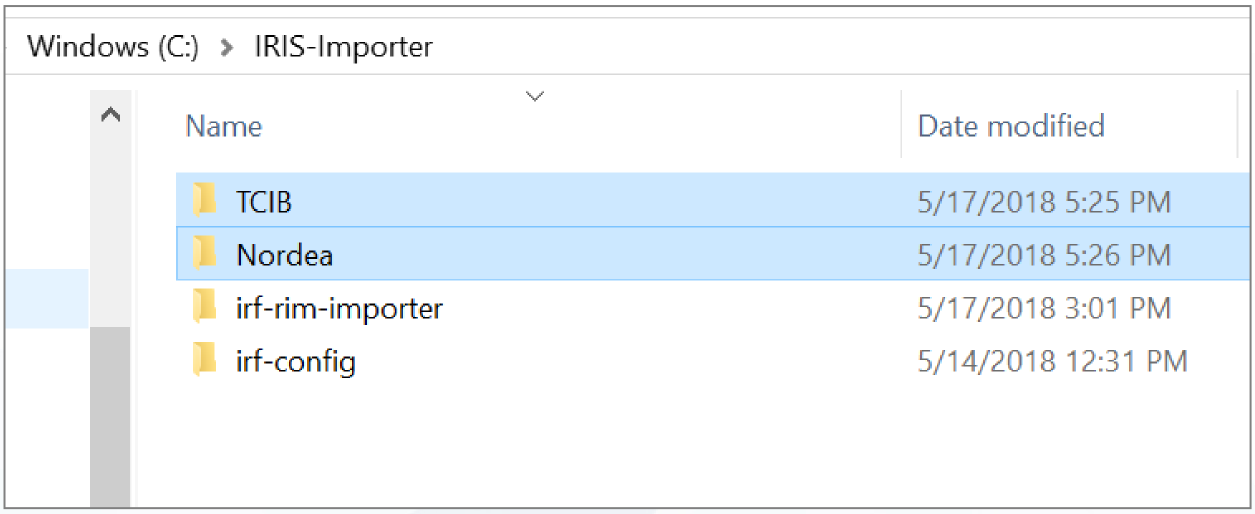
Things to remember

1. Click Download to download the latest rimimporter release package from the Temenos Maven Repository

   Link : http://maven.temenosgroup.com/content/groups/temenos/com/temenos/irf/irf-rim-importer-packager/DEV.0.0-SNAPSHOT/

2. Setup Import Configuration

a. Copy all rims to any valid folder and edit **"1.extract-apis.bat"** to change the source path.

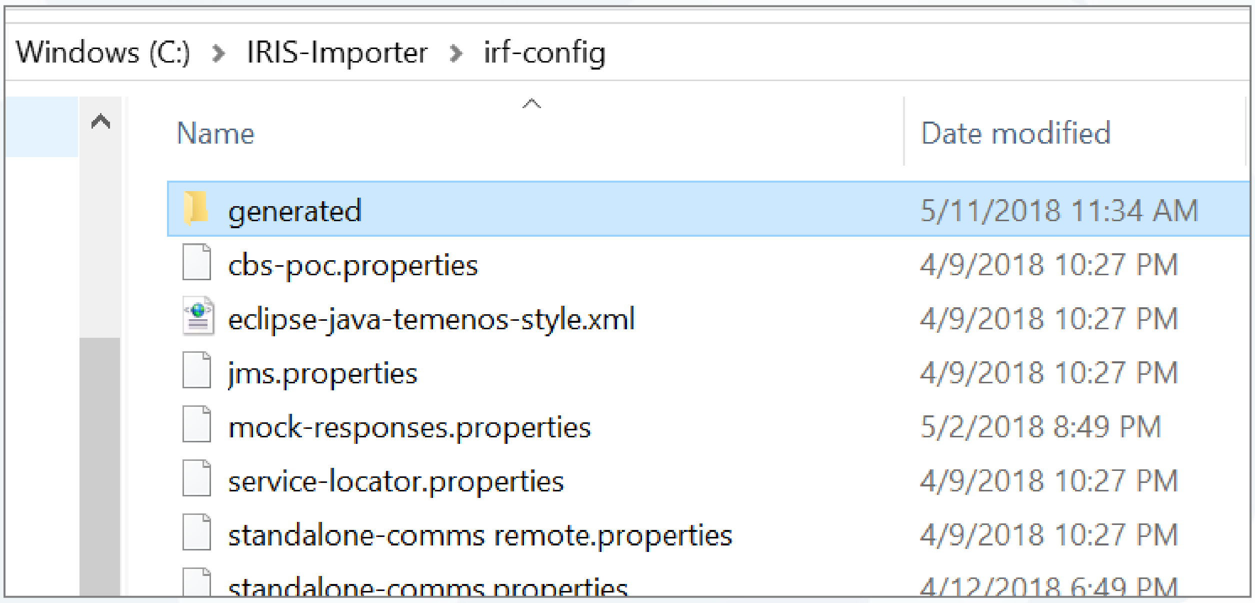


Windows (C:) > IRIS-Importer

Name	Date modified
TCIB	5/17/2018 5:25 PM
Nordea	5/17/2018 5:26 PM
irf-rim-importer	5/17/2018 3:01 PM
irf-config	5/14/2018 12:31 PM

b. Copy all corresponding generated properties files to **"..\irf-config\generated"**

Note: Needed for hand-coded or modified rims not following the IRIS1 naming standards



Windows (C:) > IRIS-Importer > irf-config

Name	Date modified
generated	5/11/2018 11:34 AM
cbs-poc.properties	4/9/2018 10:27 PM
eclipse-java-temenos-style.xml	4/9/2018 10:27 PM
jms.properties	4/9/2018 10:27 PM
mock-responses.properties	5/2/2018 8:49 PM
service-locator.properties	4/9/2018 10:27 PM
standalone-comms remote.properties	4/9/2018 10:27 PM
standalone-comms properties	4/12/2018 6:49 PM

3. T24 - Jboss is running **[Mandatory]**
4. IRF Service Container Project - Jetty is running **[Mandatory]**

Note: Refer documentation on how to create IRF Service Container Project.

5. IRF Service Project - referenced in IRF Service Container Project **[Mandatory]**

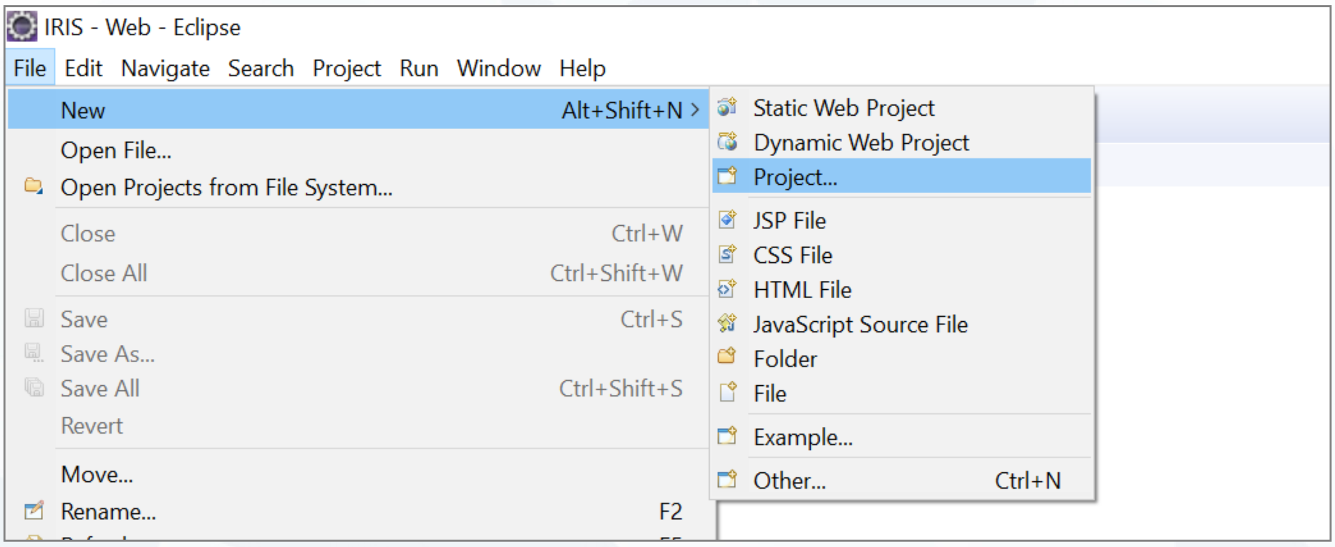
Note: Refer documentation on how to create IRF Service Project.

6. Workbench - JBoss is running **[Optional]**

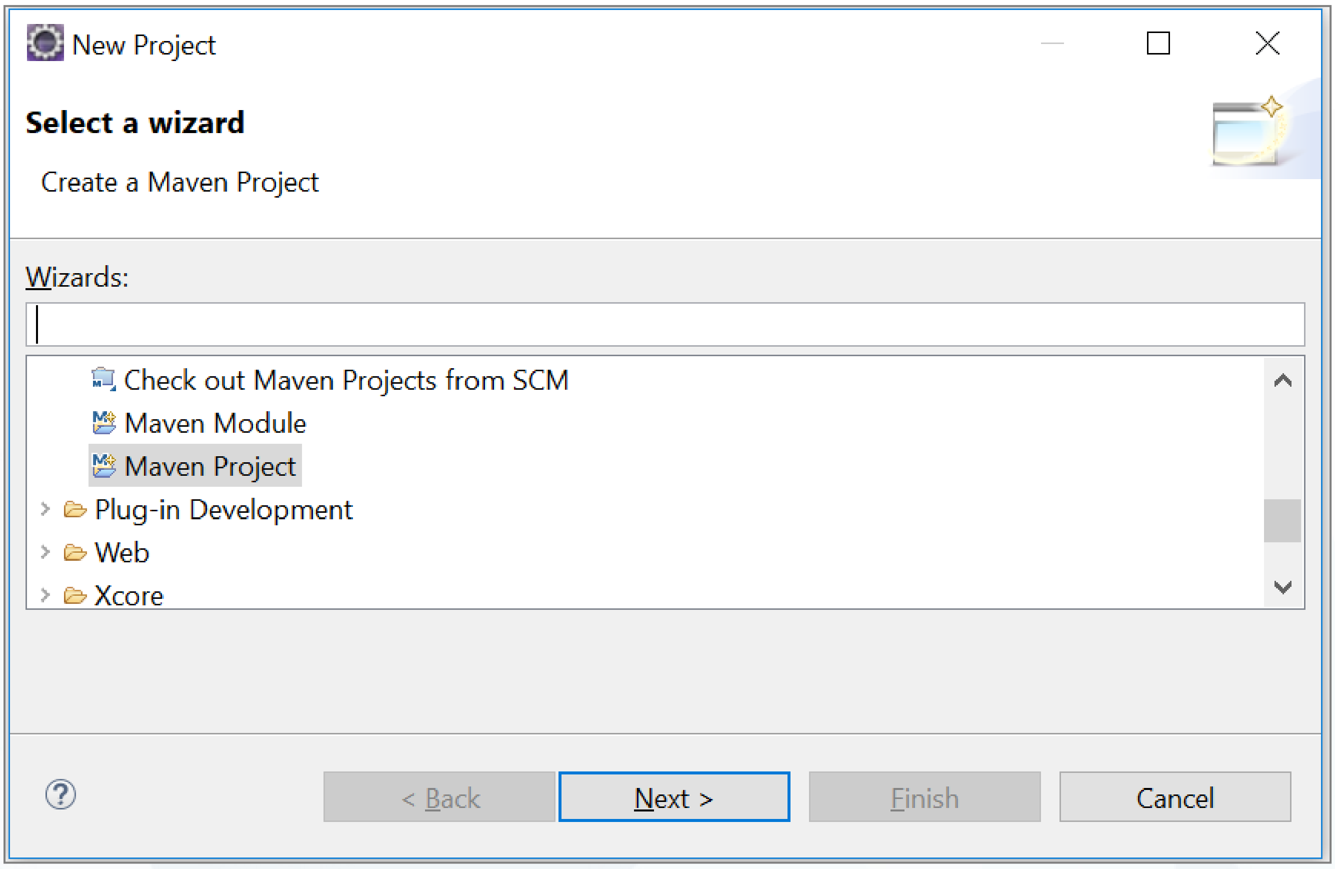
   Link: http://localhost:9089/irf-web-client/

How to create IRF Service Container Project

1. Open Eclipse or EDS
2. Click File > New Project



3. Select Maven Project
4. Click "**Next**"


New Project

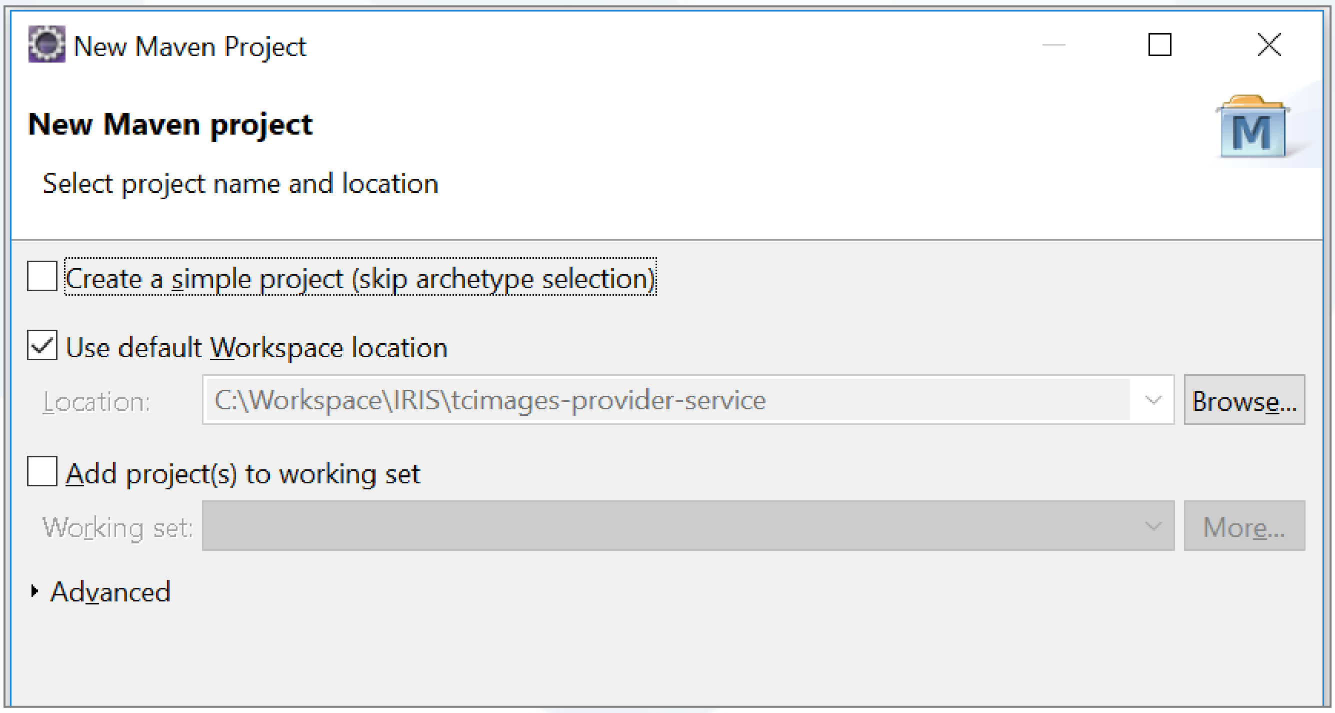
Select a wizard

Create a Maven Project

Wizards:

- Check out Maven Projects from SCM
- Maven Module
- Maven Project
- Plug-in Development
- Web
- Xcore

5. Uncheck "**Create a simple project**"
6. Click "**Next**"


New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

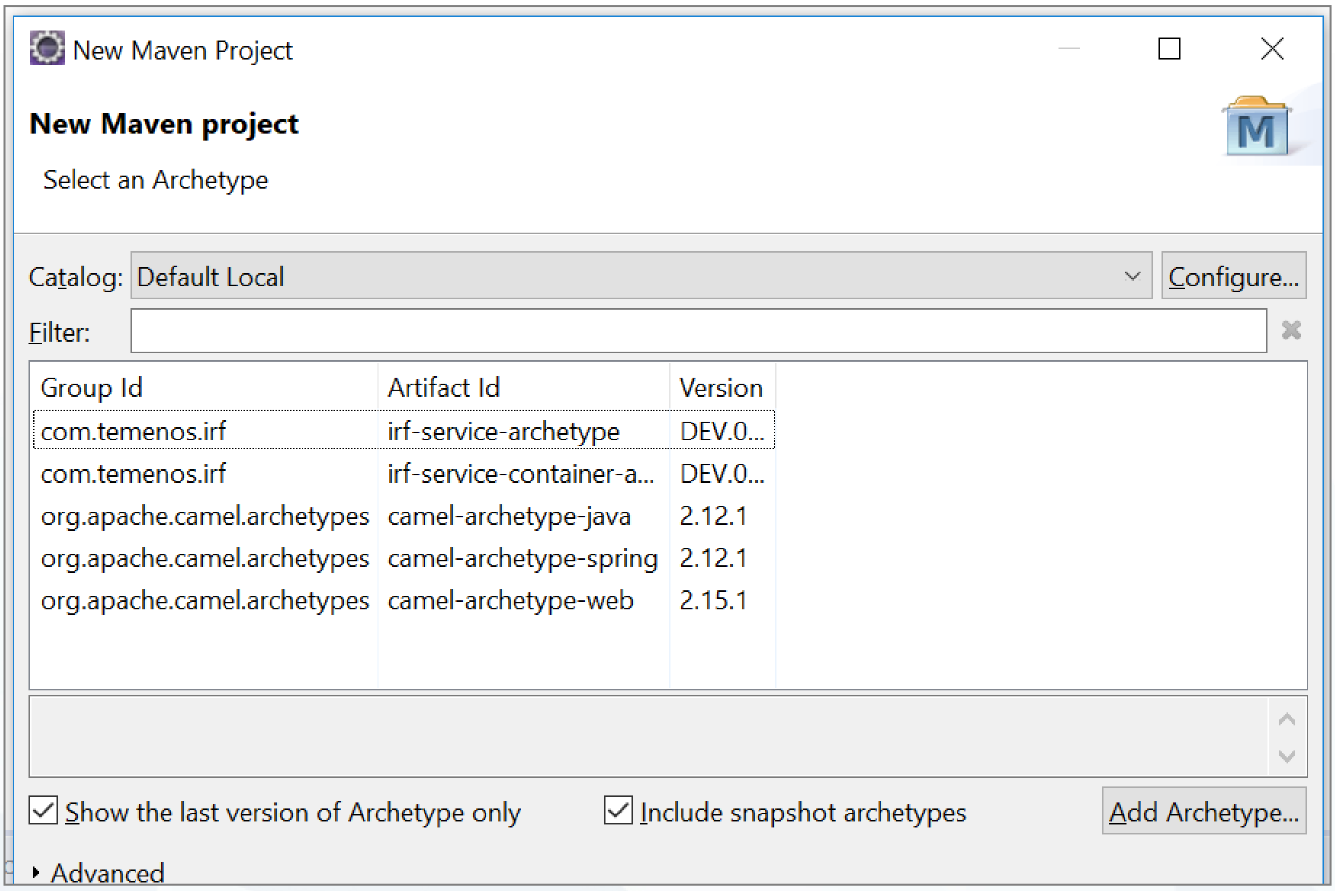
☑ Use default Workspace location

Location: C:\Workspace\IRIS\tcimages-provider-service

☐ Add project(s) to working set

Working set:

7. Check "**Include snapshot archetypes**"



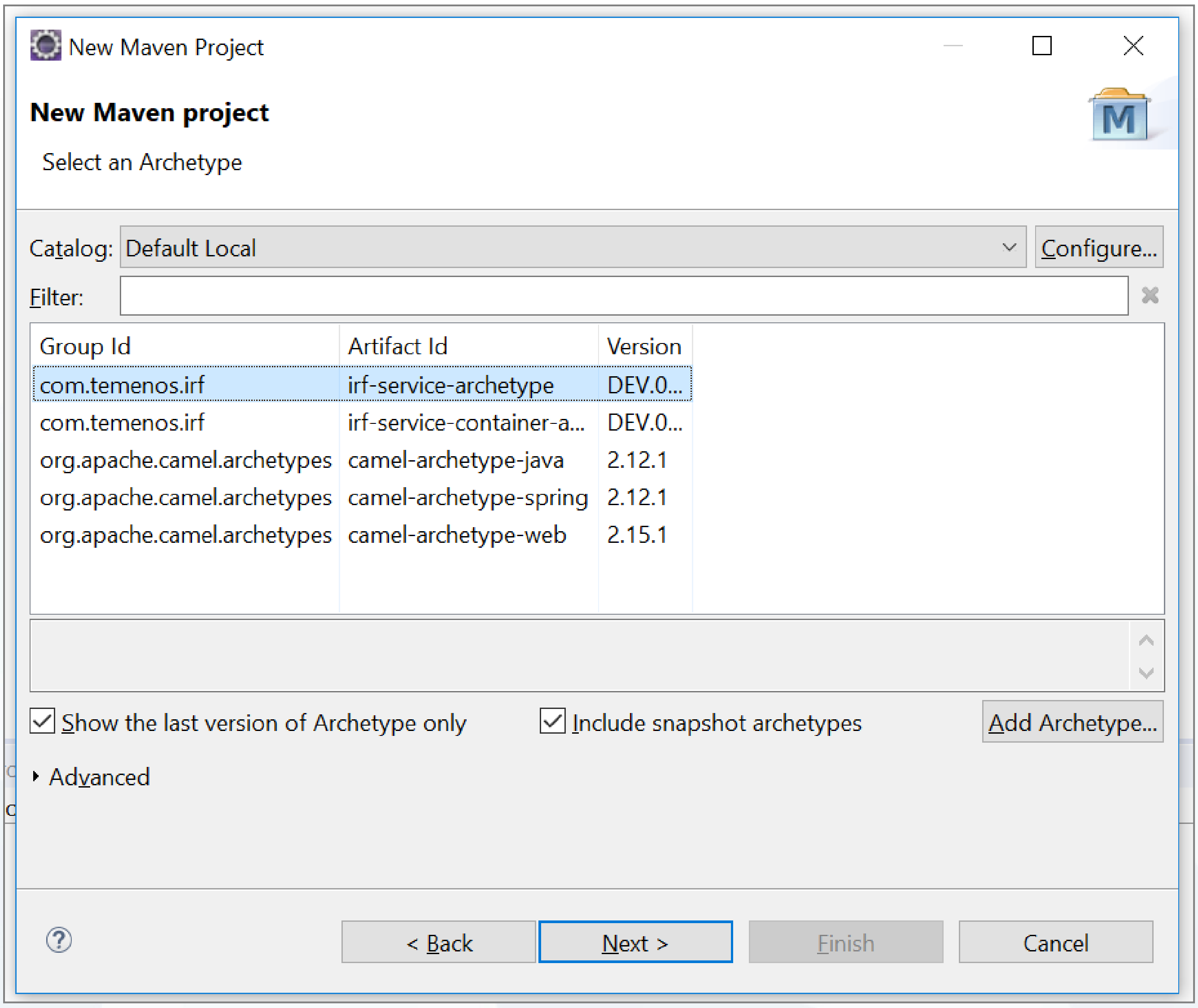
Catalog: Default Local

Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	DEV.0...
com.temenos.irf	irf-service-container-a...	DEV.0...
org.apache.camel.archetypes	camel-archetype-java	2.12.1
org.apache.camel.archetypes	camel-archetype-spring	2.12.1
org.apache.camel.archetypes	camel-archetype-web	2.15.1

☑ Show the last version of Archetype only ☑ Include snapshot archetypes

8. Select Item with Artifact Id "**irf-service-container-archetype**"
9. Click "**Next**"


New Maven Project

New Maven project

Select an Archetype

Catalog: Default Local

Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	DEV.0...
com.temenos.irf	irf-service-container-a...	DEV.0...
org.apache.camel.archetypes	camel-archetype-java	2.12.1
org.apache.camel.archetypes	camel-archetype-spring	2.12.1
org.apache.camel.archetypes	camel-archetype-web	2.15.1

☑ Show the last version of Archetype only ☑ Include snapshot archetypes

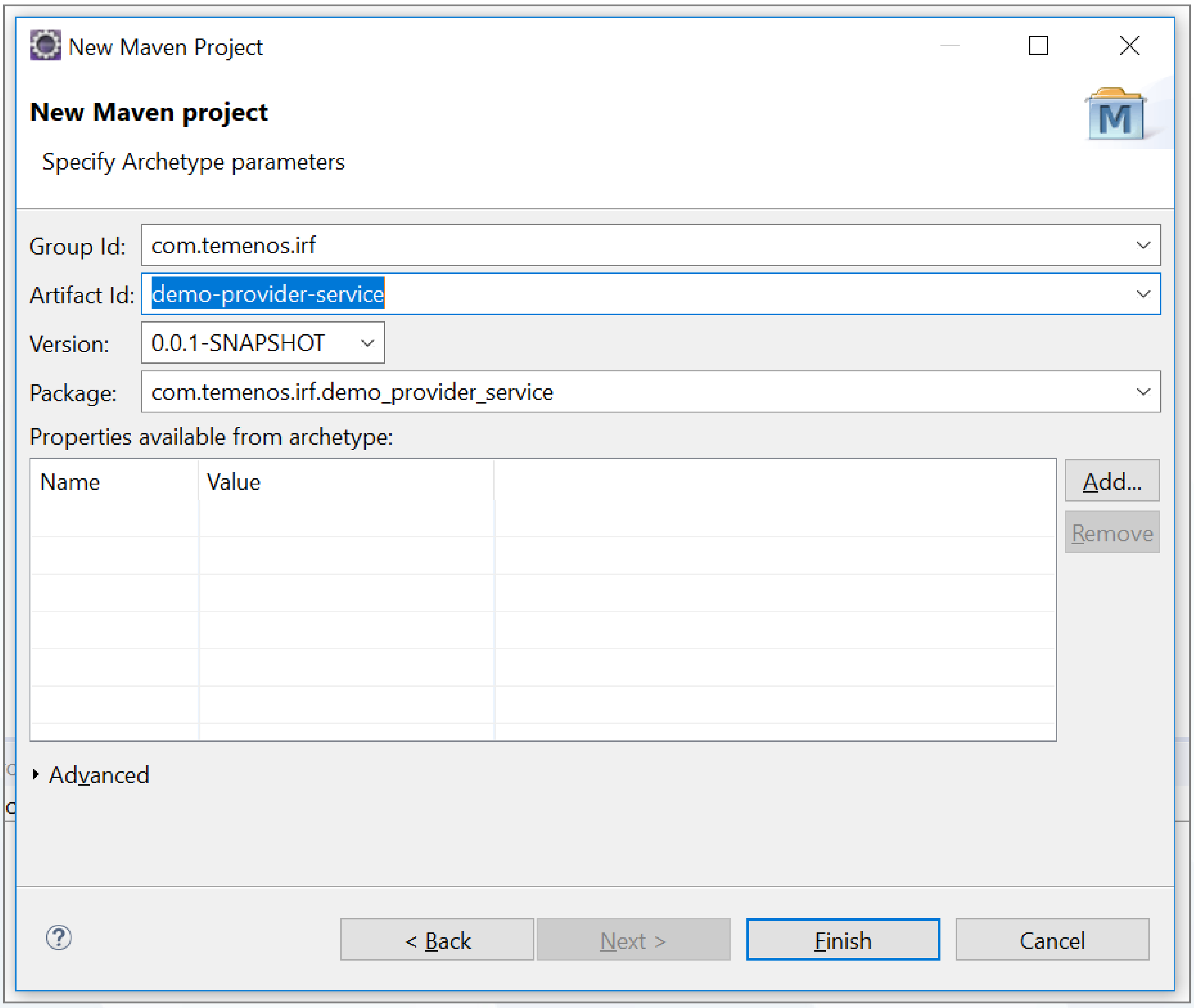
10. Enter project details

    GroupID : com.temenos.irf

    ArtifactId : demo-provider-container

    Version : 0.0.1-SNAPSHOT [Replace appropriate version]

11. Click "**Finish**"


New Maven Project

New Maven project

Specify Archetype parameters

Group Id: com.temenos.irf

Artifact Id: demo-provider-service

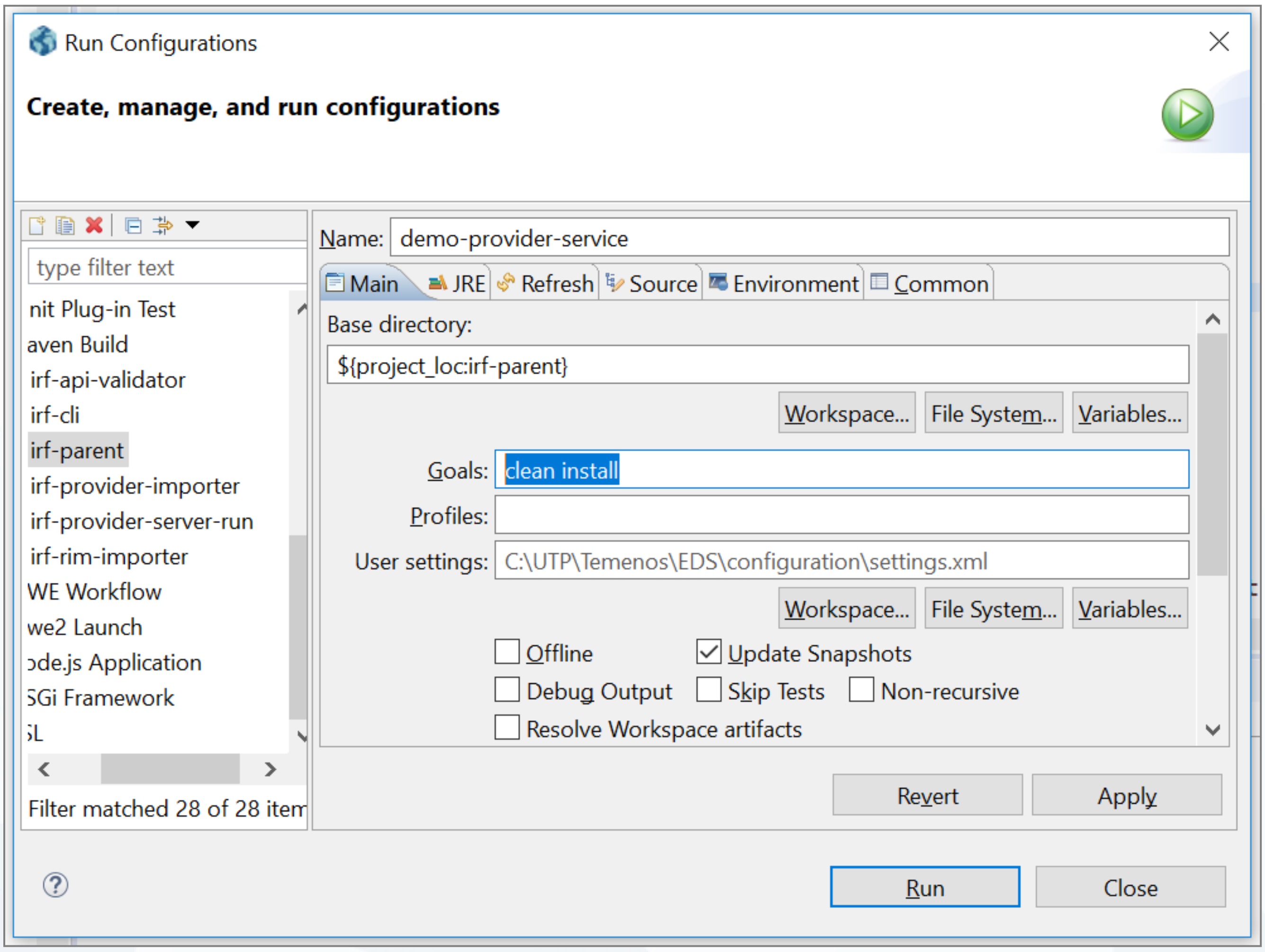
Version: 0.0.1-SNAPSHOT

Package: com.temenos.irf.demo_provider_service

Properties available from archetype:

Name	Value

12. Find the irf service container project created
13. Setup Build Configuration
    a. Select the project
    b. Right click on project
    c. Select Run As > Maven Build... [First time only]
    d. Find Configuration Window Popup
    e. Go to Goals
    f. Enter "**clean install**"
    g. Click "**Apply**" & "**Run**"


Run Configurations

Create, manage, and run configurations

type filter text

nit Plug-in Test
aven Build
irf-api-validator
irf-cli
irf-parent
irf-provider-importer
irf-provider-server-run
irf-rim-importer
WE Workflow
we2 Launch
›de.js Application
SGi Framework
›L

Filter matched 28 of 28 iterr

Name: demo-provider-service

Main | JRE | Refresh | Source | Environment | Common

Base directory: ${project_loc:irf-parent}

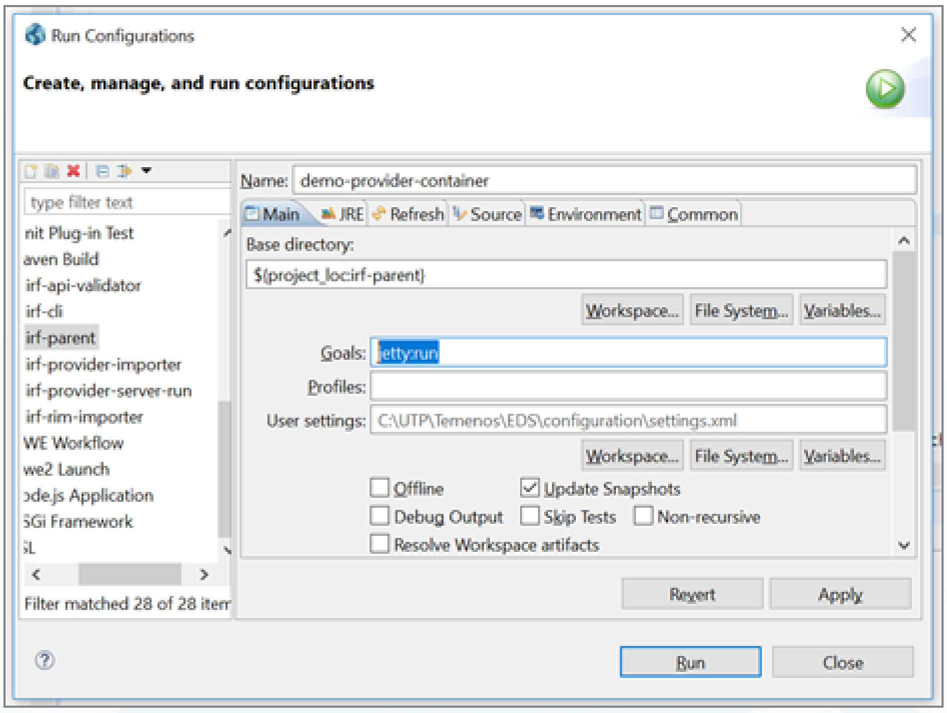
Goals: clean install

Profiles:

User settings: C:\UTP\Temenos\EDS\configuration\settings.xml

[ ] Offline [✓] Update Snapshots
[ ] Debug Output [ ] Skip Tests [ ] Non-recursive
[ ] Resolve Workspace artifacts

14. Setup Run Configuration
    a. Select the project
    b. Right click on project
    c. Select Run As > Maven Build... [First time only]
    d. Find Configuration Window Popup
    e. Go to Goals
    f. Enter "**jetty:run**"
    g. Click "**Apply**" & "**Run**"



Name: demo-provider-container

Base directory: ${project_loc:irf-parent}

Goals: jetty:run

Profiles:

User settings: C:\UTP\Temenos\EDS\configuration\settings.xml

☐ Offline ☑ Update Snapshots

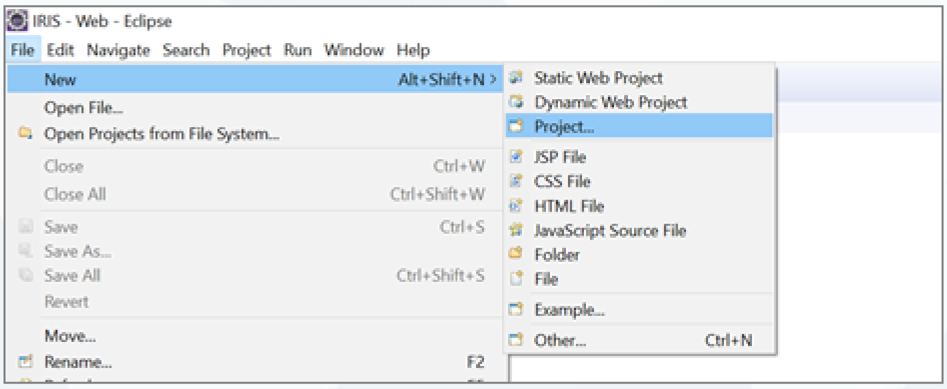
☐ Debug Output ☐ Skip Tests ☐ Non-recursive

☐ Resolve Workspace artifacts

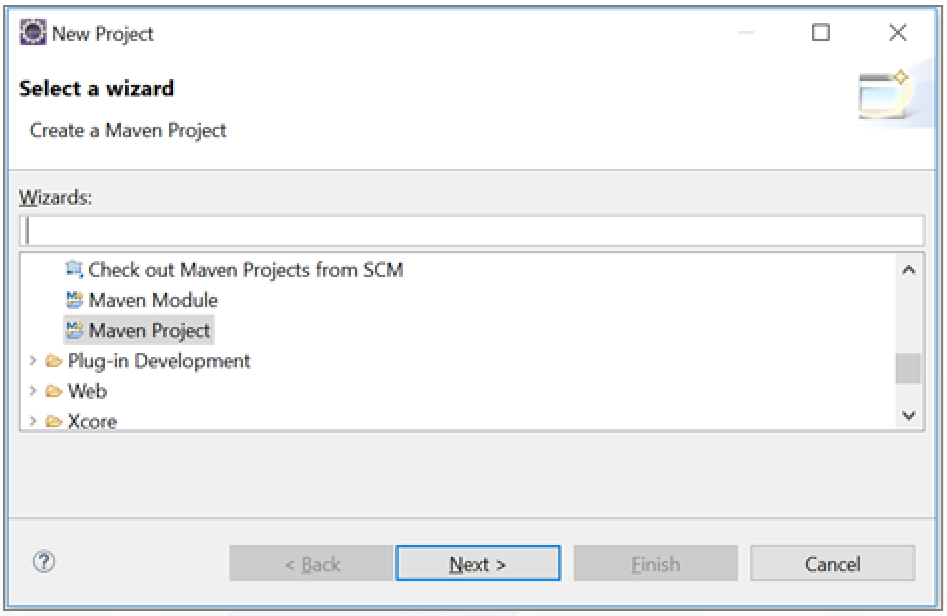
15. Right click and Select Run As > Maven Build
16. Check Eclipse or EDS Console Window for Message "**Jetty Started...**"

How to create IRF Service Project

1. Open Eclipse or EDS
2. Click File > New Project



3. Select Maven Project
4. Click "**Next**"


New Project

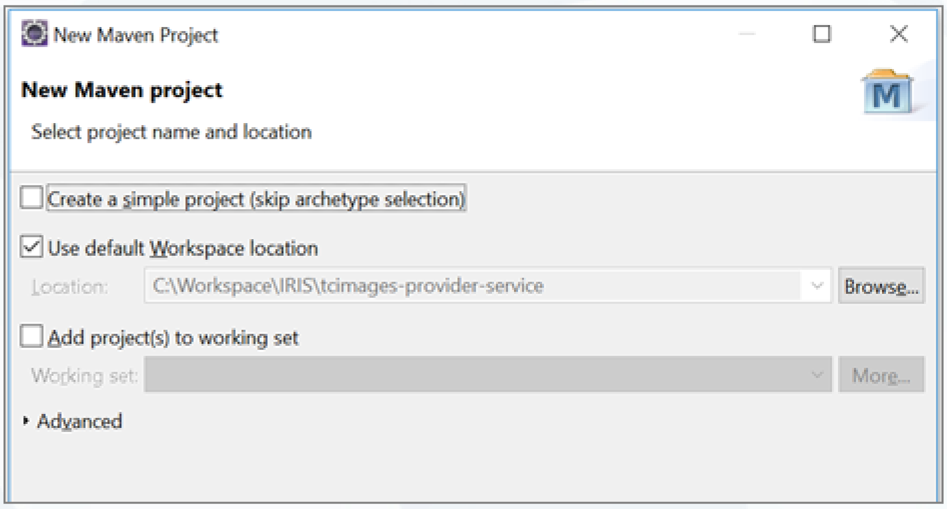
Select a wizard

Create a Maven Project

Wizards:

- Check out Maven Projects from SCM
- Maven Module
- Maven Project
- Plug-in Development
- Web
- Xcore

5. Uncheck "**Create a simple project**"
6. Click "**Next**"


New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

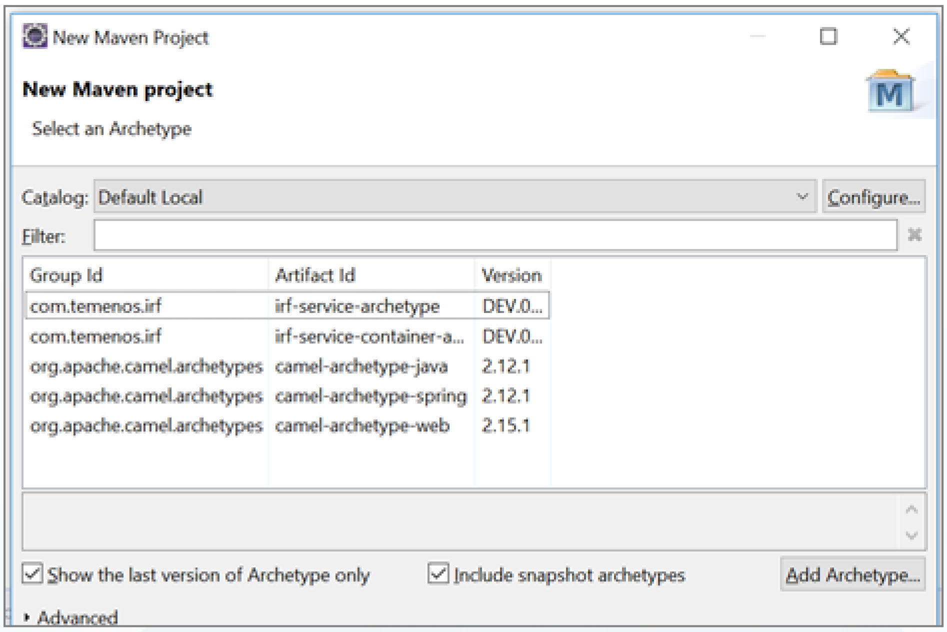
☑ Use default Workspace location

Location: C:\Workspace\IRIS\tcimages-provider-service

☐ Add project(s) to working set

Working set:

7. Check "**Include snapshot archetypes**"



New Maven Project

New Maven project

Select an Archetype

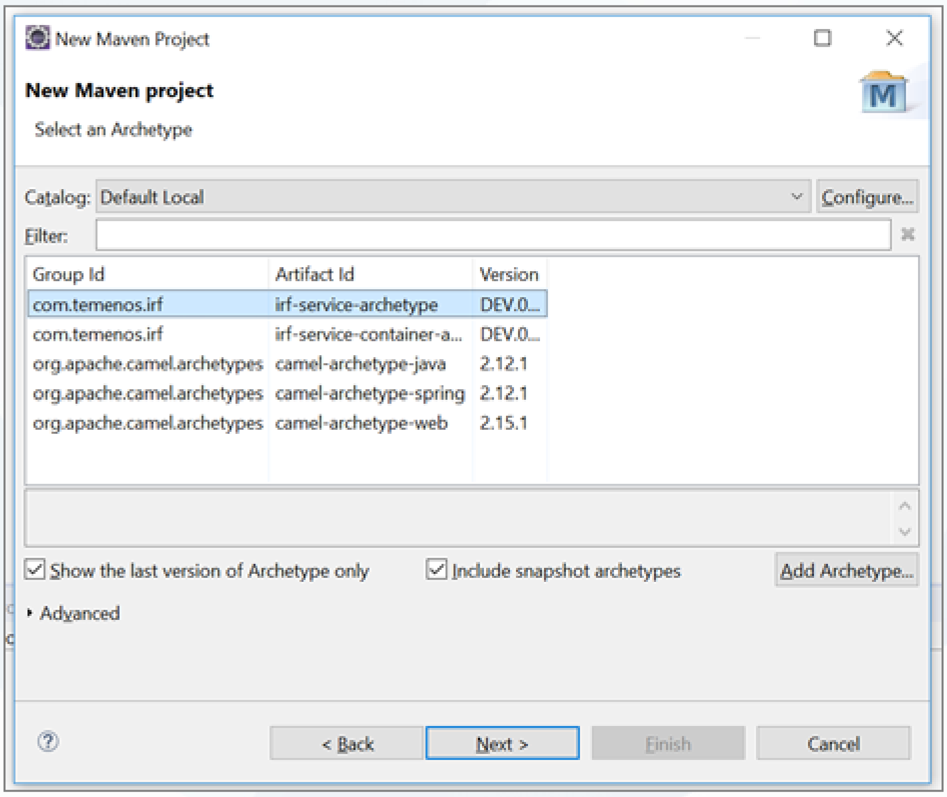
Catalog: Default Local

Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	DEV.0...
com.temenos.irf	irf-service-container-a...	DEV.0...
org.apache.camel.archetypes	camel-archetype-java	2.12.1
org.apache.camel.archetypes	camel-archetype-spring	2.12.1
org.apache.camel.archetypes	camel-archetype-web	2.15.1

☑ Show the last version of Archetype only ☑ Include snapshot archetypes

8. Select Item with Artifact Id "**irf-service-archetype**"
9. Click "**Next**"



New Maven Project

New Maven project

Select an Archetype

Catalog: Default Local

Filter:

Group Id	Artifact Id	Version
com.temenos.irf	irf-service-archetype	DEV.0...
com.temenos.irf	irf-service-container-a...	DEV.0...
org.apache.camel.archetypes	camel-archetype-java	2.12.1
org.apache.camel.archetypes	camel-archetype-spring	2.12.1
org.apache.camel.archetypes	camel-archetype-web	2.15.1

☑ Show the last version of Archetype only ☑ Include snapshot archetypes

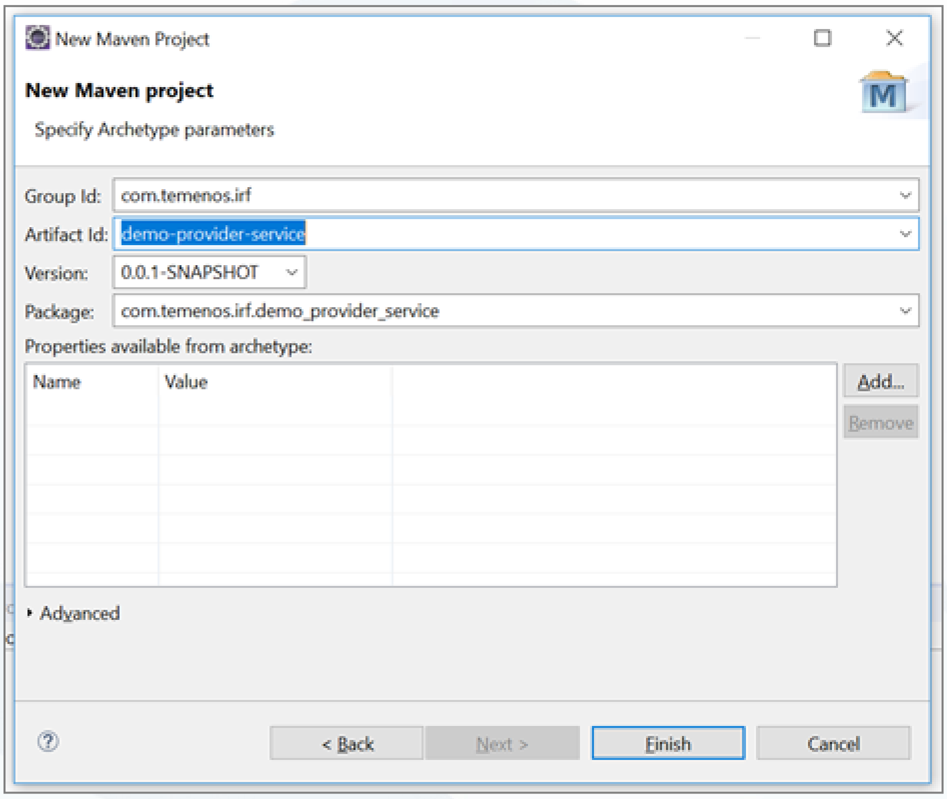
10. Enter project details

GroupID : com.temenos.irf

ArtifactId : demo-provider-service

Version : 0.0.1-SNAPSHOT [Replace appropriate version]

11. Click "**Finish**"



New Maven Project

New Maven project

Specify Archetype parameters

Group Id: com.temenos.irf

Artifact Id: demo-provider-service

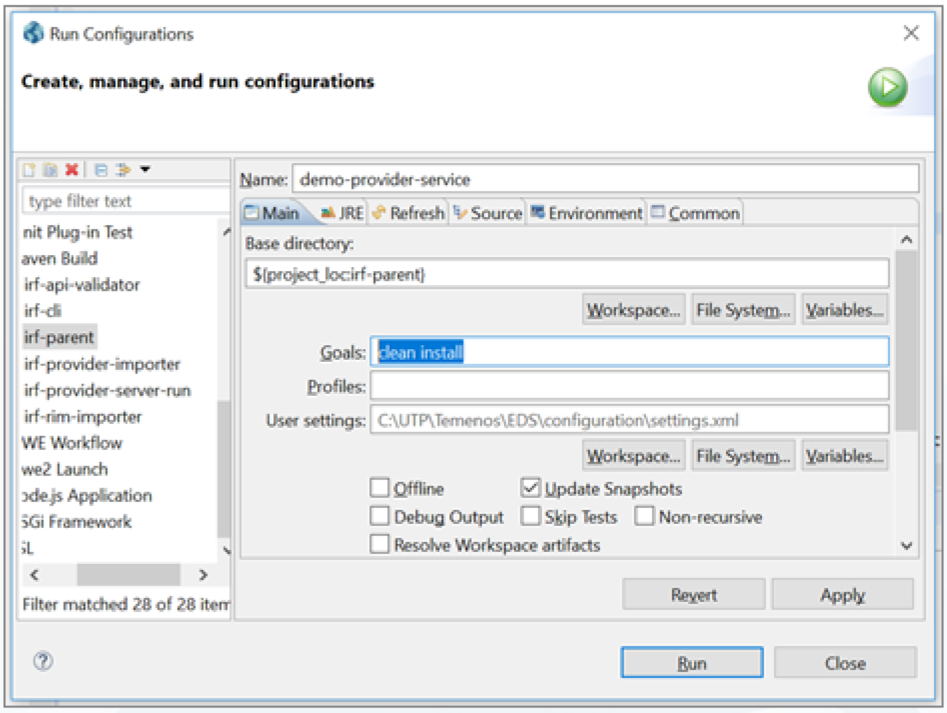
Version: 0.0.1-SNAPSHOT

Package: com.temenos.irf.demo_provider_service

Properties available from archetype:

Name	Value

12. Find the irf service container project created
13. Setup Build Configuration
    a. Select the project
    b. Right click on project
    c. Select Run As > Maven Build... [First time only]
    d. Find Configuration Window Popup
    e. Go to Goals
    f. Enter "**clean install**"
    g. Click **Apply** and **Run**.



Run Configurations

Create, manage, and run configurations

Name: demo-provider-service

Base directory: ${project_loc:irf-parent}

Goals: clean install

Profiles:

User settings: C:\UTP\Temenos\EDS\configuration\settings.xml

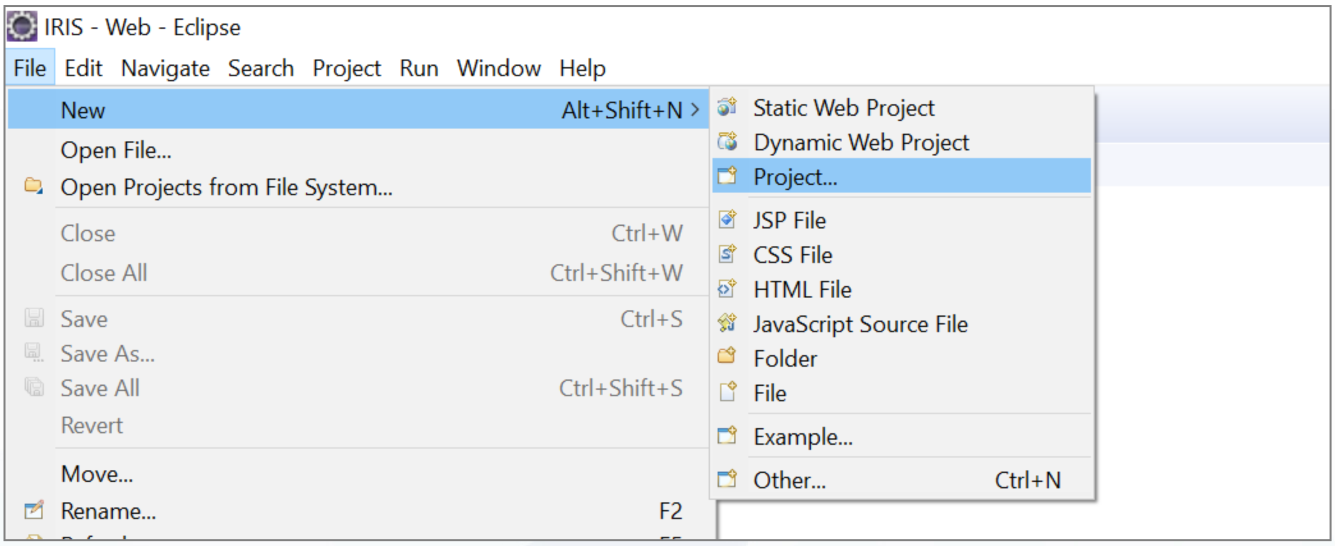
- [ ] Offline
- [✓] Update Snapshots
- [ ] Debug Output
- [ ] Skip Tests
- [ ] Non-recursive
- [ ] Resolve Workspace artifacts

Filter matched 28 of 28 items

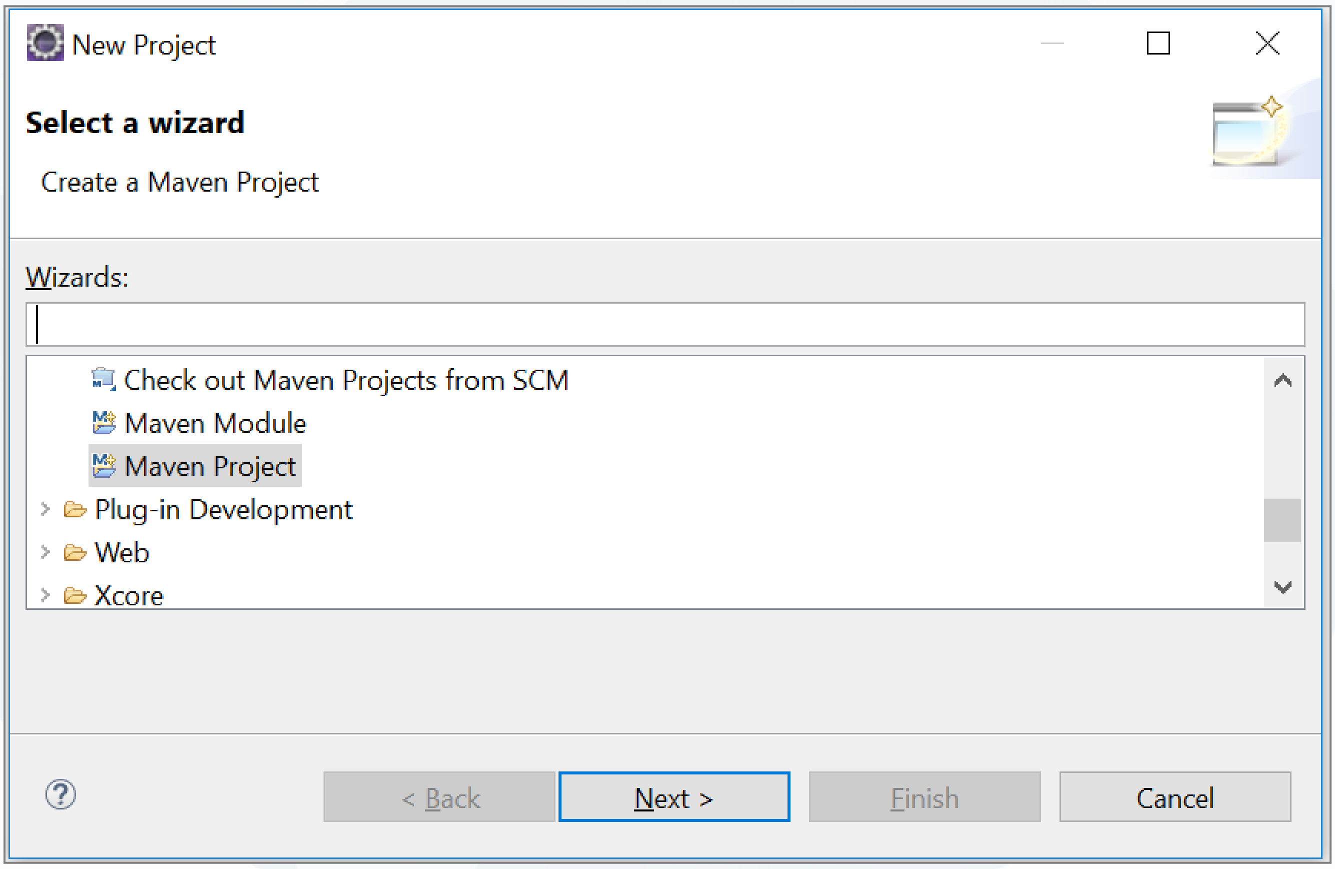
14. Reference IRF Service Project in IRF Service Container Project
    a. Open POM XML
    b. Copy the Artifact Details
    c. Go to IRF Service Contained Project
    d. Open POM XML
    e. Add an artifact reference under references
    f. Paste the details
        i. GroupId
        ii. ArtefactId
        iii. Version
    g. Save POM XML (Both Service and container Projects)
15. Right Click and Select Run As > Maven Build
16. Check Eclipse or EDS Console Window for Message "**Jetty Restarted...**"

How to add IRF Archetypes

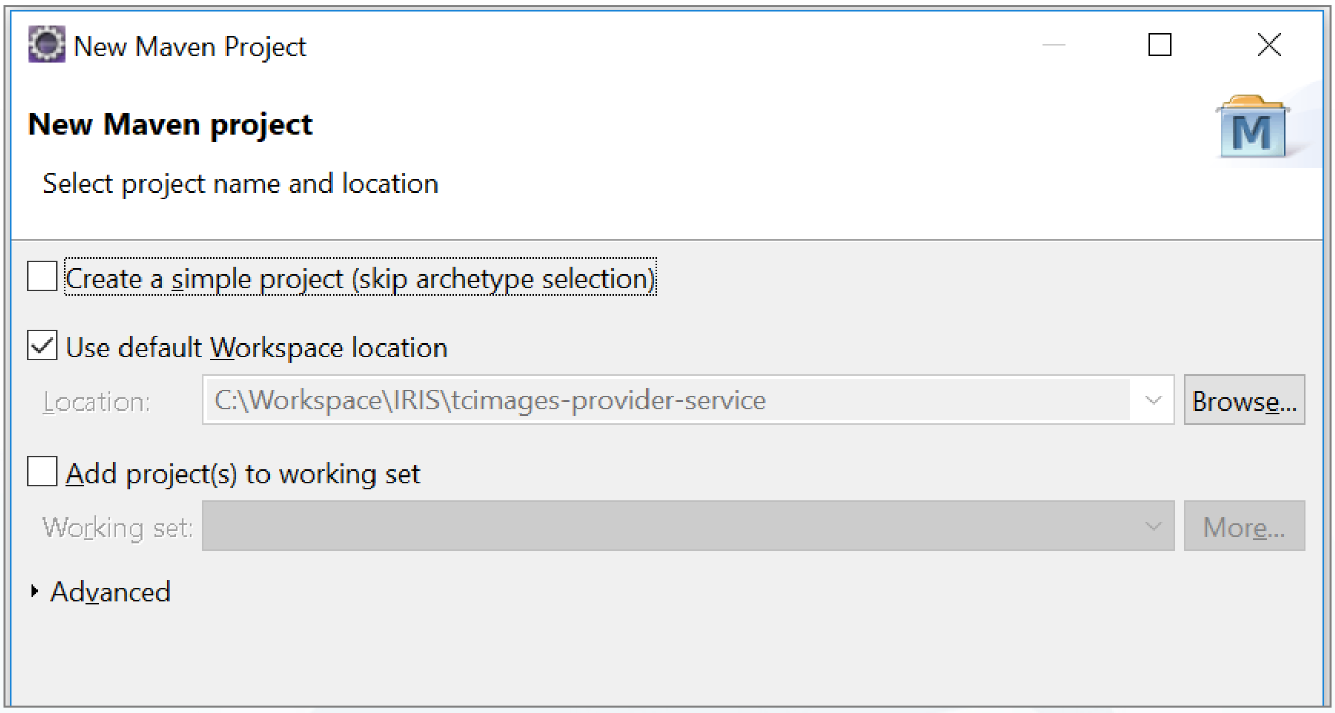
1. Open Eclipse or EDS
2. Click File > New Project



3. Select Maven Project
4. Click **"Next"**



5. Uncheck **"Create a simple project"**
6. Click **"Next"**



New Maven project

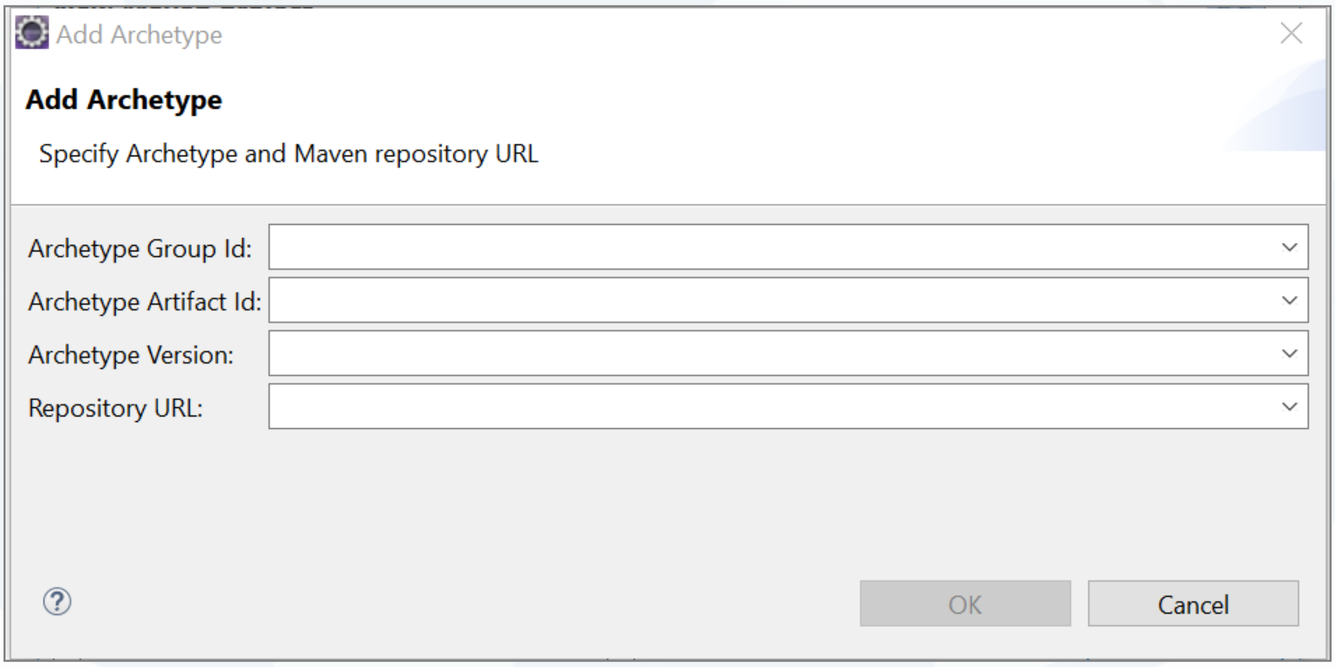
Select project name and location

☐ Create a simple project (skip archetype selection)

☑ Use default Workspace location

Location: C:\Workspace\IRIS\tcimages-provider-service

7. Click **"Add Archetype..."**



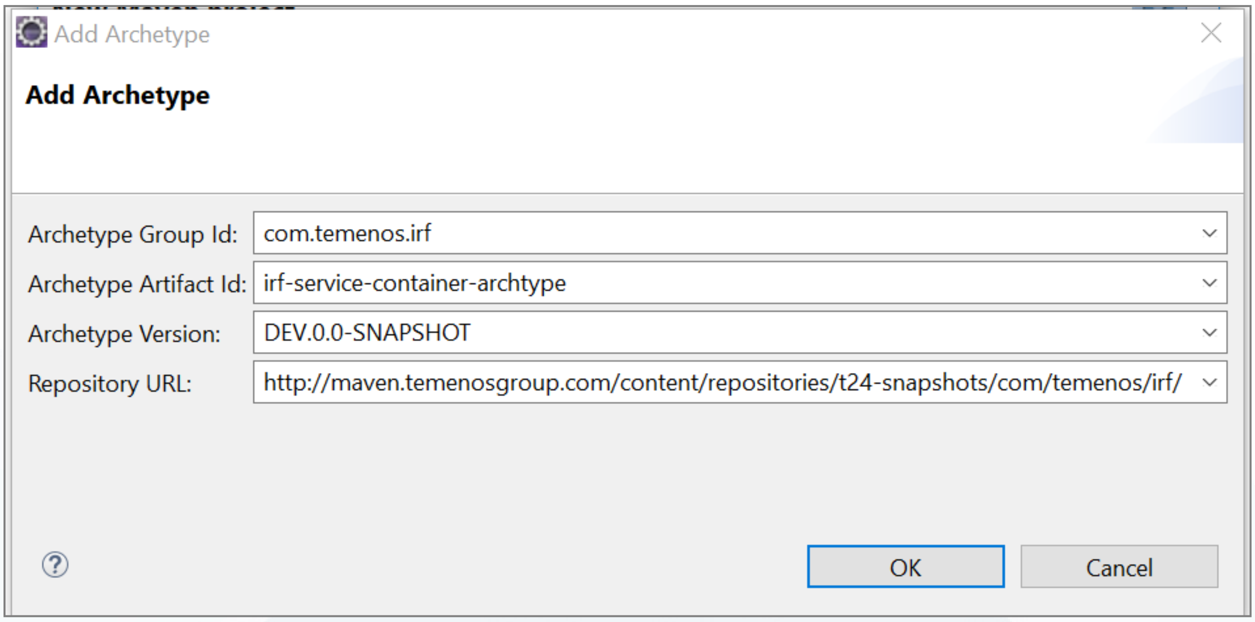
Add Archetype

Specify Archetype and Maven repository URL

Archetype Group Id:	
Archetype Artifact Id:	
Archetype Version:	
Repository URL:	

8. Provide Archetype details

a. IRF Service Container Project Archtype



Archetype Group Id:	com.temenos.irf
Archetype Artifact Id:	irf-service-container-archtype
Archetype Version:	DEV.0.0-SNAPSHOT
Repository URL:	http://maven.temenosgroup.com/content/repositories/t24-snapshots/com/temenos/irf/

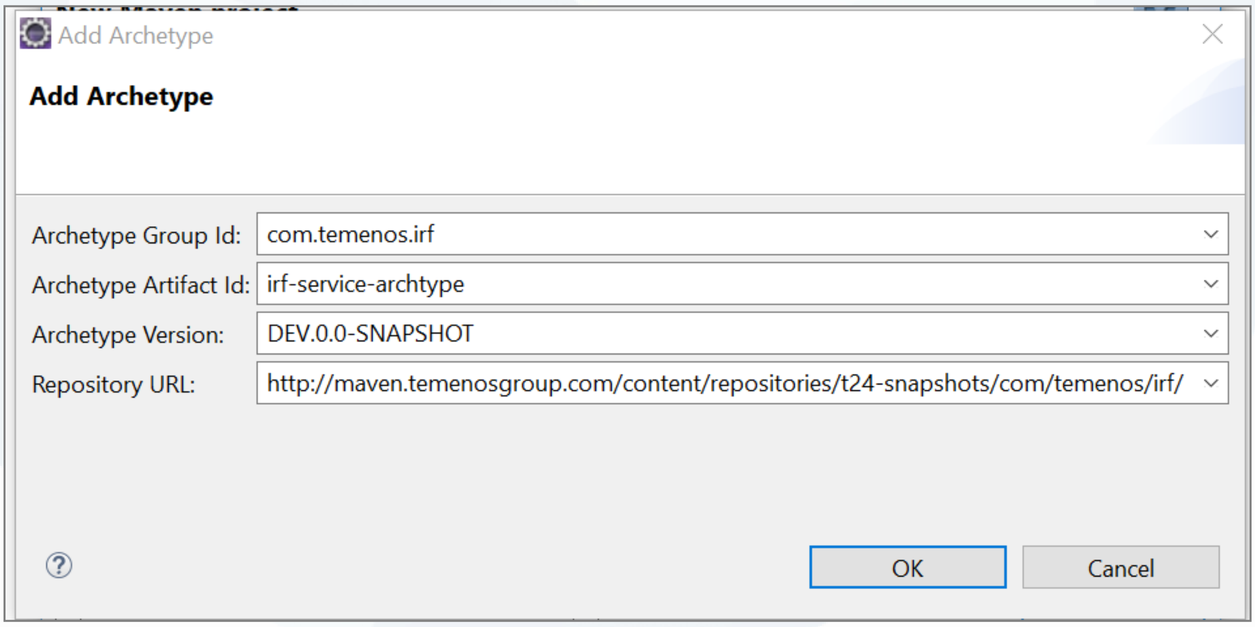
GroupID : com.temenos.irf

ArtifactId : irf-service-container-archtype

Version : DEV.0.0-SNAPSHOT [Replace appropriate version]

URL: http://maven.temenosgroup.com/content/repositories/t24-snapshots/com/temenos/irf/

b. IRF Service Project Archtype



Archetype Group Id:	com.temenos.irf
Archetype Artifact Id:	irf-service-archtype
Archetype Version:	DEV.0.0-SNAPSHOT
Repository URL:	http://maven.temenosgroup.com/content/repositories/t24-snapshots/com/temenos/irf/

GroupID : com.temenos.irf

ArtifactId : irf-service-archetype

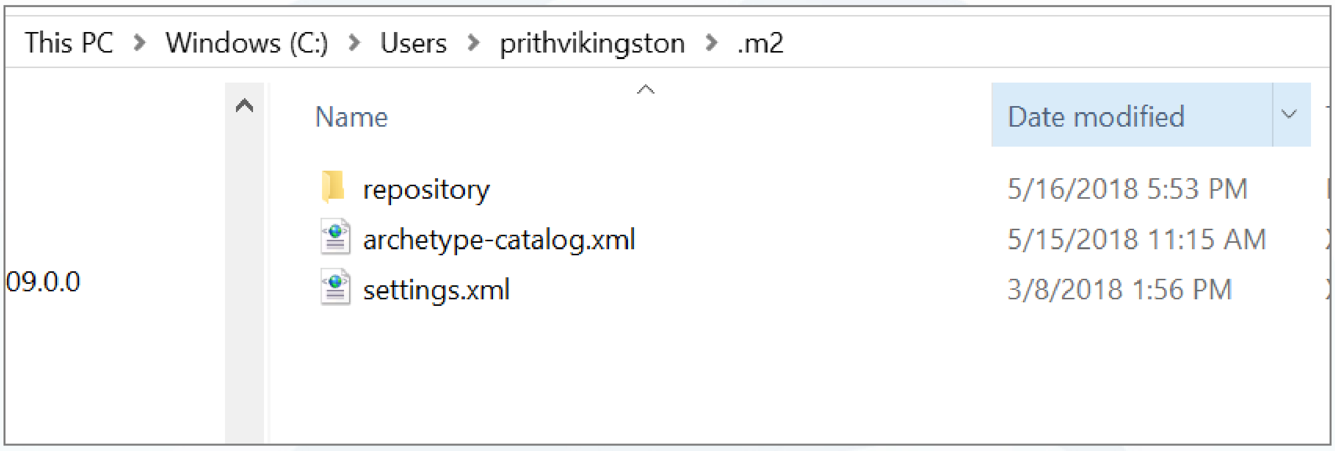
Version : DEV.0.0-SNAPSHOT [Replace appropriate version]

URL : http://maven.temenosgroup.com/content/repositories/t24-snapshots/com/temenos/irf/

9. Click "OK" to add archetype

Alternatively

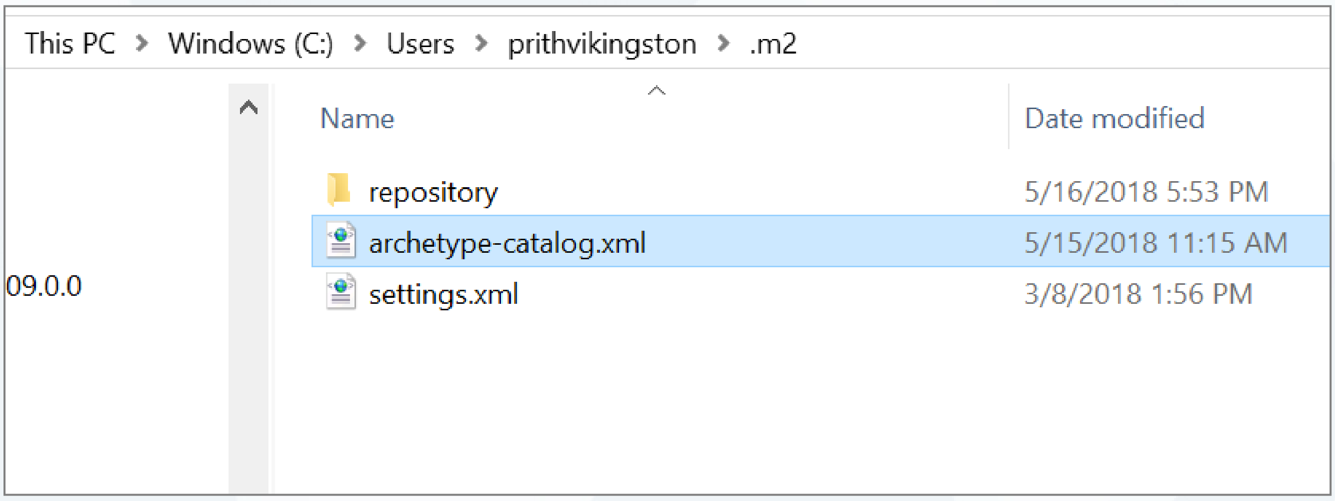
1. >Browse to **"C:\Users\<UserName>\.m2"** directory of maven



This PC > Windows (C:) > Users > prithvikingston > .m2

Name	Date modified
repository	5/16/2018 5:53 PM
archetype-catalog.xml	5/15/2018 11:15 AM
settings.xml	3/8/2018 1:56 PM

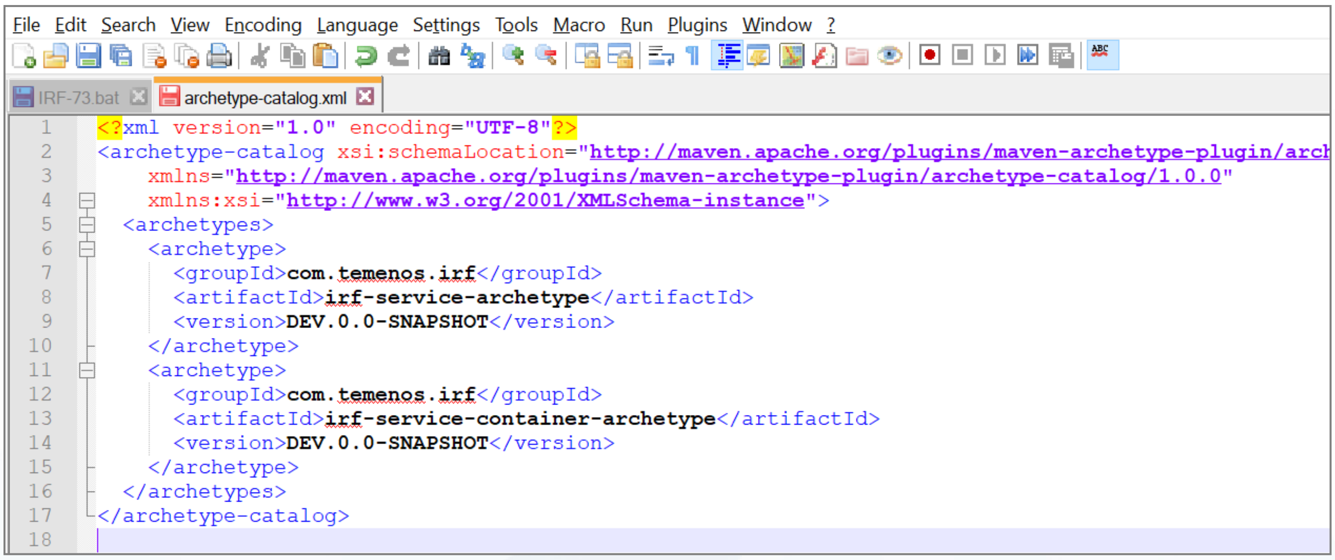
2. Find and Open **"archetype-catalog.xml"**



This PC > Windows (C:) > Users > prithvikingston > .m2

Name	Date modified
repository	5/16/2018 5:53 PM
archetype-catalog.xml	5/15/2018 11:15 AM
settings.xml	3/8/2018 1:56 PM

3. Add the below details between <archetypes> ... </archetypes>


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
IRF-73.bat
archetype-catalog.xml

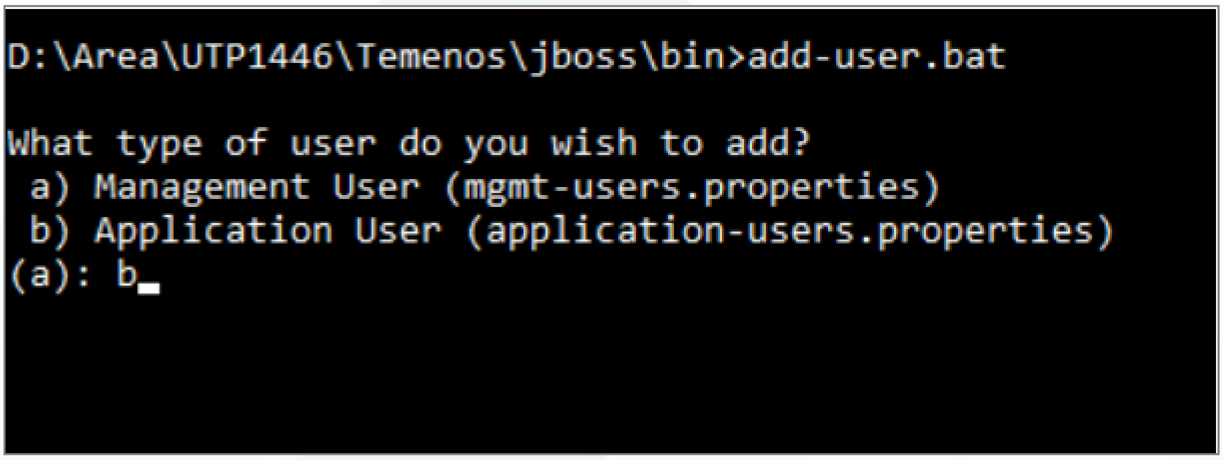
```
<?xml version="1.0" encoding="UTF-8"?>
<archetype-catalog xsi:schemaLocation="http://maven.apache.org/plugins/maven-archetype-plugin/arch
    xmlns="http://maven.apache.org/plugins/maven-archetype-plugin/archetype-catalog/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <archetypes>
    <archetype>
      <groupId>com.temenos.irf</groupId>
      <artifactId>irf-service-archetype</artifactId>
      <version>DEV.0.0-SNAPSHOT</version>
    </archetype>
    <archetype>
      <groupId>com.temenos.irf</groupId>
      <artifactId>irf-service-container-archetype</artifactId>
      <version>DEV.0.0-SNAPSHOT</version>
    </archetype>
  </archetypes>
</archetype-catalog>
```

```
<archetype>

<groupId>com.temenos.irf</groupId>

<artifactId>irf-service-container-archetype</artifactId>

<version>DEV.0.0-SNAPSHOT</version>

</archetype>

<archetype>

<groupId>com.temenos.irf</groupId>

<artifactId>irf-service-archetype</artifactId>

<version>DEV.0.0-SNAPSHOT</version>

</archetype>
```

4. Save the file

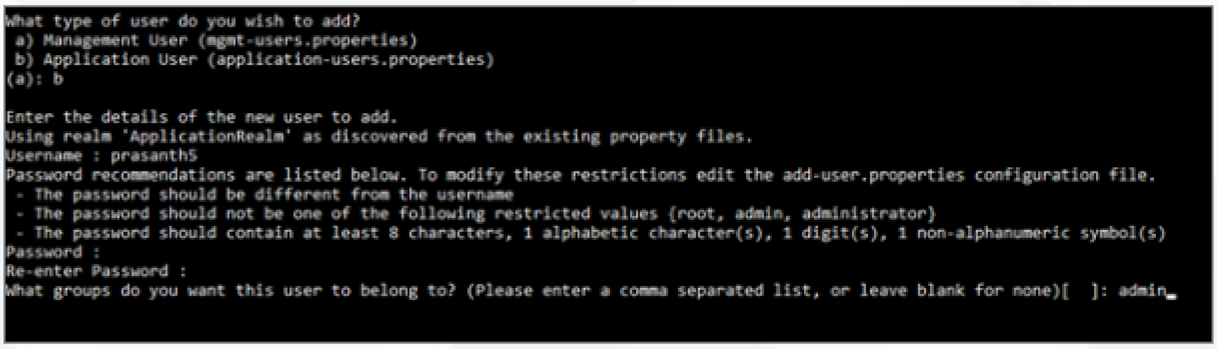
How to configure RimImporter to communicate Remote T24 Server

1. Create a new "Application User" in JBoss (Remote Server i.e T24Server)
   a. Go to <JBOSS_HOME>/bin/add-user.bat



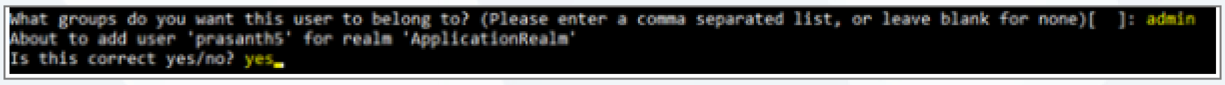
```
D:\Area\UTP1446\Temenos\jboss\bin>add-user.bat

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): b_
```

   b. Select option '**b**' to start creating an "**Application User**"
   c. Enter your username and password as per instruction



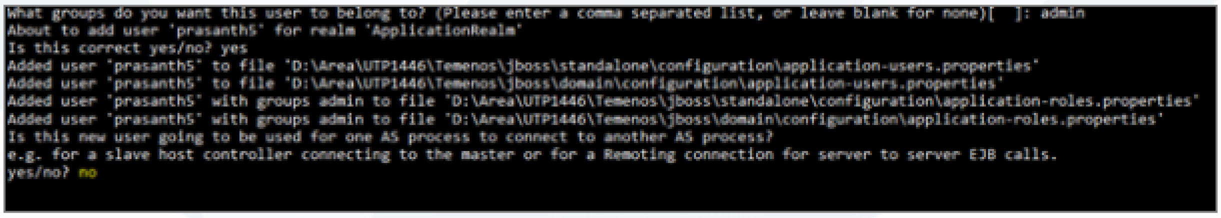
```
What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): b

Enter the details of the new user to add.
Using realm 'ApplicationRealm' as discovered from the existing property files.
Username : prasanth5
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]: admin_
```

   d. Enter your group as "**admin**"



```
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]: admin
About to add user 'prasanth5' for realm 'ApplicationRealm'
Is this correct yes/no? yes_
```

   e. Enter "**No**" for server-server EJB call



```
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]: admin
About to add user 'prasanth5' for realm 'ApplicationRealm'
Is this correct yes/no? yes
Added user 'prasanth5' to file 'D:\Area\UTP1446\Temenos\jboss\standalone\configuration\application-users.properties'
Added user 'prasanth5' to file 'D:\Area\UTP1446\Temenos\jboss\domain\configuration\application-users.properties'
Added user 'prasanth5' with groups admin to file 'D:\Area\UTP1446\Temenos\jboss\standalone\configuration\application-roles.properties'
Added user 'prasanth5' with groups admin to file 'D:\Area\UTP1446\Temenos\jboss\domain\configuration\application-roles.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
yes/no? no
```

2. Configure the RimImporter Importer.properties

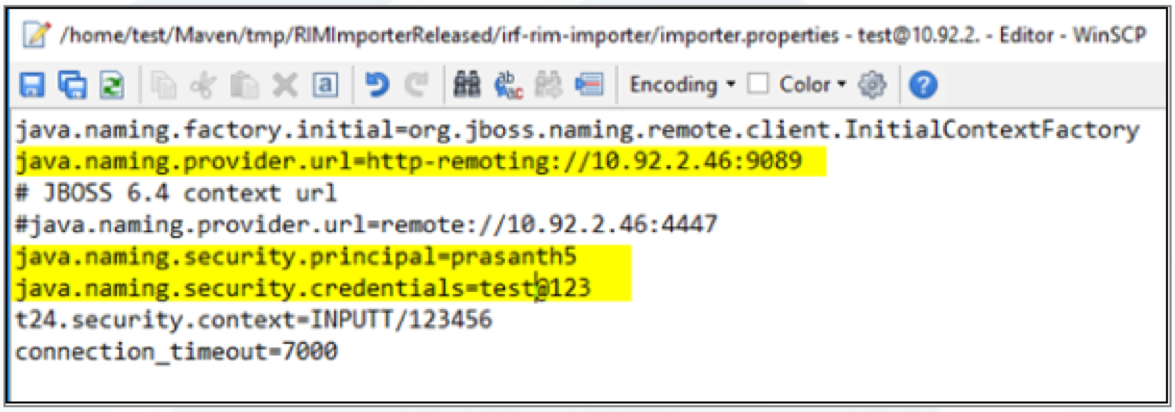
a. Go to irf-rim-importer directory

b. Open "importer.properties" configuration file

c. Update the below section details

   i. url in java.naming.provider

   ii. username and password in java.naming.security


/home/test/Maven/tmp/RIMImporterReleased/irf-rim-importer/importer.properties - test@10.92.2. - Editor - WinSCP
Encoding
Color

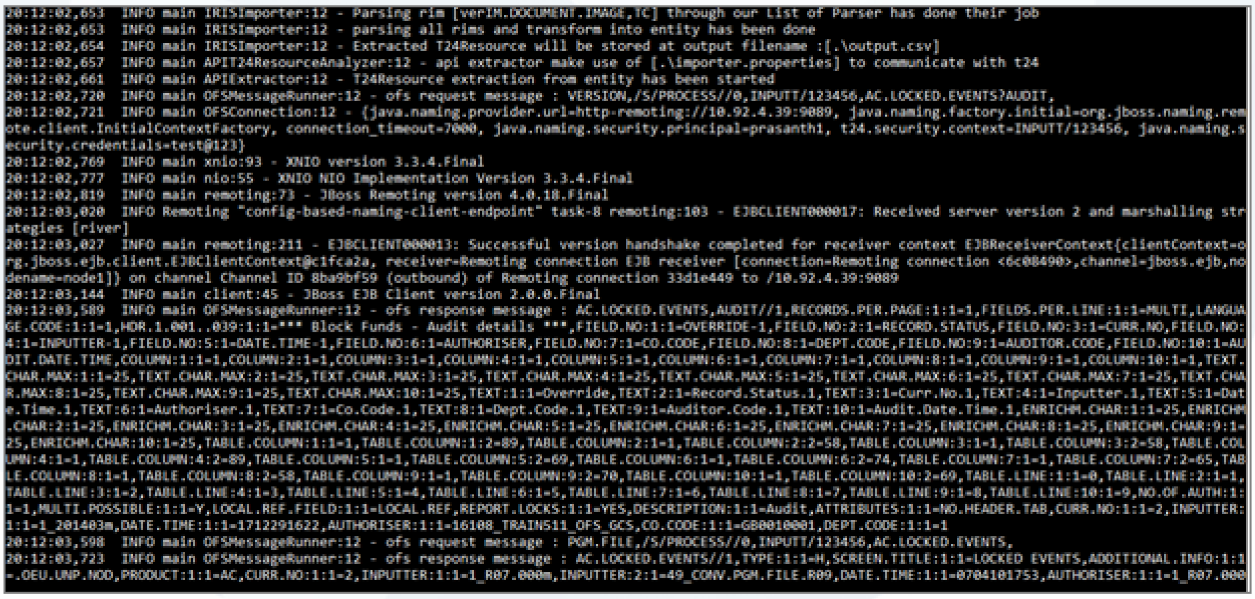
```
java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextFactory
java.naming.provider.url=http-remoting://10.92.2.46:9089
# JBOSS 6.4 context url
#java.naming.provider.url=remote://10.92.2.46:4447
java.naming.security.principal=prasanth5
java.naming.security.credentials=test@123
t24.security.context=INPUTT/123456
connection_timeout=7000
```

3. Verify the Configuration and Setup

   a. Go to irf-rim-importer directory

   b. Run 1.extract-api.bat to extract the API's from configured RIMS or Location

   c. Ensure that No Errors on Connection

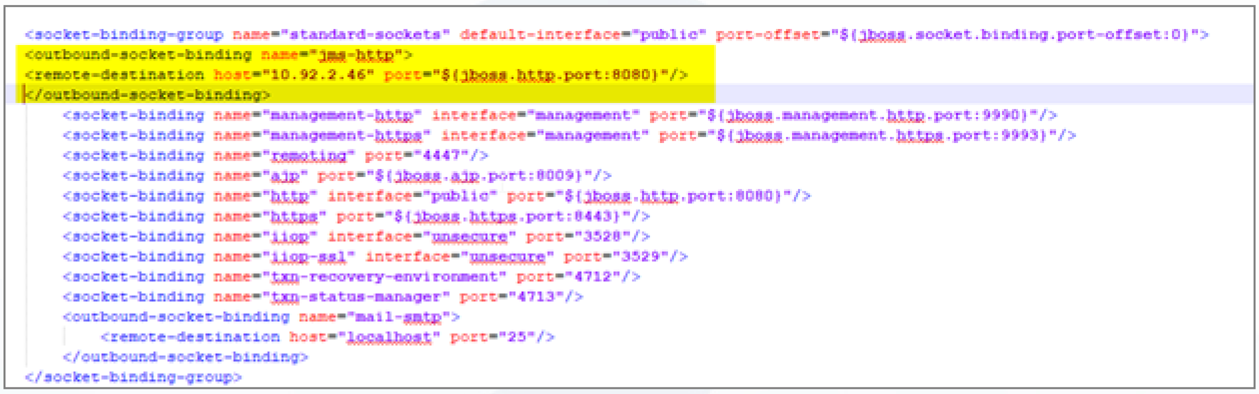


```
20:12:02,653  INFO main IRISImporter:12 - Parsing rim [verIM.DOCUMENT.IMAGE,TC] through our List of Parser has done their job
20:12:02,653  INFO main IRISImporter:12 - parsing all rims and transform into entity has been done
20:12:02,654  INFO main IRISImporter:12 - Extracted T24Resource will be stored at output filename :[.\output.csv]
20:12:02,657  INFO main APIT24ResourceAnalyzer:12 - api extractor make use of [.\importer.properties] to communicate with t24
20:12:02,661  INFO main APIExtractor:12 - T24Resource extraction from entity has been started
20:12:02,720  INFO main OFSMessageRunner:12 - ofs request message : VERSION,/S/PROCESS//0,INPUTT/123456,AC.LOCKED.EVENTS?AUDIT,
20:12:02,721  INFO main OFSConnection:12 - {java.naming.provider.url=http-remoting://10.92.4.39:9089, java.naming.factory.initial=org.jboss.naming.rem
ote.client.InitialContextFactory, connection_timeout=7000, java.naming.security.principal=prasanth1, t24.security.context=INPUTT/123456, java.naming.s
ecurity.credentials=test@123}
20:12:02,769  INFO main xnio:93 - XNIO version 3.3.4.Final
20:12:02,777  INFO main nio:55 - XNIO NIO Implementation Version 3.3.4.Final
20:12:02,819  INFO main remoting:73 - JBoss Remoting version 4.0.18.Final
20:12:03,020  INFO Remoting "config-based-naming-client-endpoint" task-8 remoting:103 - EJBCLIENT000017: Received server version 2 and marshalling str
ategies [river]
20:12:03,027  INFO main remoting:211 - EJBCLIENT000013: Successful version handshake completed for receiver context EJBReceiverContext{clientContext=o
rg.jboss.ejb.client.EJBClientContext@c1fca2a, receiver=Remoting connection EJB receiver [connection=Remoting connection <6c08490>,channel=jboss.ejb,no
dename=node1]} on channel Channel ID 8ba9bf59 (outbound) of Remoting connection 33d1e449 to /10.92.4.39:9089
20:12:03,144  INFO main client:45 - JBoss EJB Client version 2.0.0.Final
20:12:03,589  INFO main OFSMessageRunner:12 - ofs response message : AC.LOCKED.EVENTS,AUDIT//1,RECORDS.PER.PAGE:1:1=1,FIELDS.PER.LINE:1:1=MULTI,LANGUA
GE.CODE:1:1=1,HDR.1.001..039:1:1=*** Block Funds - Audit details ***,FIELD.NO:1:1=OVERRIDE-1,FIELD.NO:2:1=RECORD.STATUS,FIELD.NO:3:1=CURR.NO,FIELD.NO:
4:1=INPUTTER-1,FIELD.NO:5:1=DATE.TIME-1,FIELD.NO:6:1=AUTHORISER,FIELD.NO:7:1=CO.CODE,FIELD.NO:8:1=DEPT.CODE,FIELD.NO:9:1=AUDITOR.CODE,FIELD.NO:10:1=AU
DIT.DATE.TIME,COLUMN:1:1=1,COLUMN:2:1=1,COLUMN:3:1=1,COLUMN:4:1=1,COLUMN:5:1=1,COLUMN:6:1=1,COLUMN:7:1=1,COLUMN:8:1=1,COLUMN:9:1=1,COLUMN:10:1=1,TEXT.
CHAR.MAX:1:1=25,TEXT.CHAR.MAX:2:1=25,TEXT.CHAR.MAX:3:1=25,TEXT.CHAR.MAX:4:1=25,TEXT.CHAR.MAX:5:1=25,TEXT.CHAR.MAX:6:1=25,TEXT.CHAR.MAX:7:1=25,TEXT.CHA
R.MAX:8:1=25,TEXT.CHAR.MAX:9:1=25,TEXT.CHAR.MAX:10:1=25,TEXT:1:1=Override,TEXT:2:1=Record.Status.1,TEXT:3:1=Curr.No.1,TEXT:4:1=Inputter.1,TEXT:5:1=Dat
e.Time.1,TEXT:6:1=Authoriser.1,TEXT:7:1=Co.Code.1,TEXT:8:1=Dept.Code.1,TEXT:9:1=Auditor.Code.1,TEXT:10:1=Audit.Date.Time.1,ENRICHM.CHAR:1:1=25,ENRICHM
.CHAR:2:1=25,ENRICHM.CHAR:3:1=25,ENRICHM.CHAR:4:1=25,ENRICHM.CHAR:5:1=25,ENRICHM.CHAR:6:1=25,ENRICHM.CHAR:7:1=25,ENRICHM.CHAR:8:1=25,ENRICHM.CHAR:9:1=
25,ENRICHM.CHAR:10:1=25,TABLE.COLUMN:1:1=1,TABLE.COLUMN:1:2=89,TABLE.COLUMN:2:1=1,TABLE.COLUMN:2:2=58,TABLE.COLUMN:3:1=1,TABLE.COLUMN:3:2=58,TABLE.COL
UMN:4:1=1,TABLE.COLUMN:4:2=89,TABLE.COLUMN:5:1=1,TABLE.COLUMN:5:2=69,TABLE.COLUMN:6:1=1,TABLE.COLUMN:6:2=74,TABLE.COLUMN:7:1=1,TABLE.COLUMN:7:2=65,TAB
LE.COLUMN:8:1=1,TABLE.COLUMN:8:2=58,TABLE.COLUMN:9:1=1,TABLE.COLUMN:9:2=70,TABLE.COLUMN:10:1=1,TABLE.COLUMN:10:2=69,TABLE.LINE:1:1=0,TABLE.LINE:2:1=1,
TABLE.LINE:3:1=2,TABLE.LINE:4:1=3,TABLE.LINE:5:1=4,TABLE.LINE:6:1=5,TABLE.LINE:7:1=6,TABLE.LINE:8:1=7,TABLE.LINE:9:1=8,TABLE.LINE:10:1=9,NO.OF.AUTH:1:
1=1,MULTI.POSSIBLE:1:1=Y,LOCAL.REF.FIELD:1:1=LOCAL.REF,REPORT.LOCKS:1:1=YES,DESCRIPTION:1:1=Audit,ATTRIBUTES:1:1=NO.HEADER.TAB,CURR.NO:1:1=2,INPUTTER:
1:1=1_201403m,DATE.TIME:1:1=1712291622,AUTHORISER:1:1=16108_TRAIN511_OFS_GCS,CO.CODE:1:1=GB0010001,DEPT.CODE:1:1=1
20:12:03,598  INFO main OFSMessageRunner:12 - ofs request message : PGM.FILE,/S/PROCESS//0,INPUTT/123456,AC.LOCKED.EVENTS,
20:12:03,723  INFO main OFSMessageRunner:12 - ofs response message : AC.LOCKED.EVENTS//1,TYPE:1:1=H,SCREEN.TITLE:1:1=LOCKED EVENTS,ADDITIONAL.INFO:1:1
=.OEU.UNP.NOD,PRODUCT:1:1=AC,CURR.NO:1:1=2,INPUTTER:1:1=1_R07.000m,INPUTTER:2:1=49_CONV.PGM.FILE.R09,DATE.TIME:1:1=0704101753,AUTHORISER:1:1=1_R07.000
```

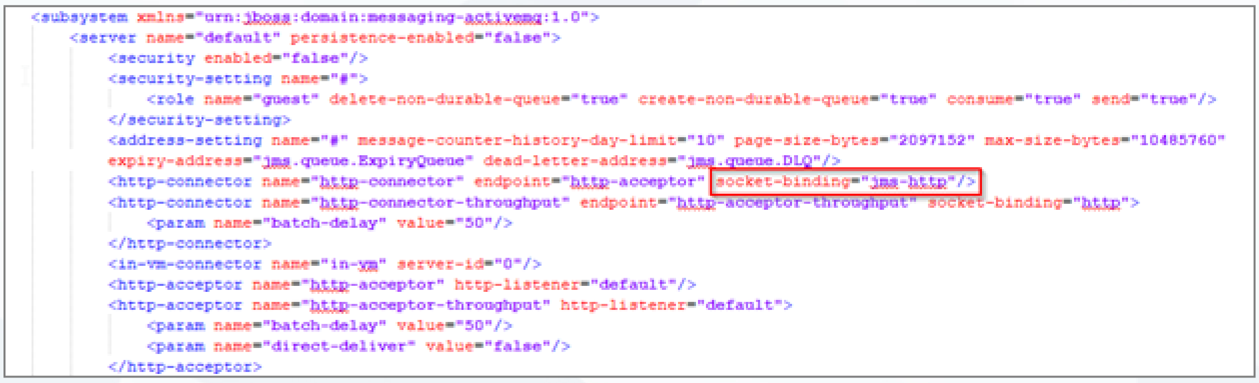
4. Troubleshoot after the configuration (Just in case of cross platform machines)

Issue : ActiveMQNotConnectedException[errorType=NOT_CONNECTED message=AMQ119007: Cannot connect to server(s). Tried with all available servers.]

Solution: Add outbound socket with actual IP and mention the binding name on http-connector



```xml
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
<outbound-socket-binding name="jms-http">
<remote-destination host="10.92.2.46" port="${jboss.http.port:8080}"/>
</outbound-socket-binding>
    <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9993}"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" interface="public" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```



```xml
<subsystem xmlns="urn:jboss:domain:messaging-activemq:1.0">
    <server name="default" persistence-enabled="false">
        <security enabled="false"/>
        <security-setting name="#">
            <role name="guest" delete-non-durable-queue="true" create-non-durable-queue="true" consume="true" send="true"/>
        </security-setting>
        <address-setting name="#" message-counter-history-day-limit="10" page-size-bytes="2097152" max-size-bytes="10485760"
        expiry-address="jms.queue.ExpiryQueue" dead-letter-address="jms.queue.DLQ"/>
        <http-connector name="http-connector" endpoint="http-acceptor" socket-binding="jms-http"/>
        <http-connector name="http-connector-throughput" endpoint="http-acceptor-throughput" socket-binding="http">
            <param name="batch-delay" value="50"/>
        </http-connector>
        <in-vm-connector name="in-vm" server-id="0"/>
        <http-acceptor name="http-acceptor" http-listener="default"/>
        <http-acceptor name="http-acceptor-throughput" http-listener="default">
            <param name="batch-delay" value="50"/>
            <param name="direct-deliver" value="false"/>
        </http-acceptor>
```

RIMImporter Configuration Files

1. **importer.properties**

   The file has all configuration required to setup the rim importer utility. The properties file contains the connection details required to establish connection with JBoss and T24Server. Below are some of the connection parameters

   #Connection Credentials

   java.naming.security.principal=INPUTT

   java.naming.security.credentials=123456

   t24.security.context=INPUTT/123456

   #Connection Timeout

   connection_timeout=7000

> Note: By default this file might not be available and will be picked using classpath resource. In case you differ from the default connection, it can be edited by copying the

above details and creating a file with importer.properties with updated details and save. The file shall and must be located in the same directory as the batch files.

2. **domain.properties**

   The file is used to categorize application list into domains like party, retail, holdings, etc. The domains are used to follow the convention of IRISX of URL resource pattern of \api\version\domain\resource\{property}\subresource\{property}. Ensure that all the applications are classified into respective domains so that the url formation is done in compliance to the IRISX conventions meeting REST Standards. Another key information that the domains are also used to refer the right vocabulary file for its resource and properties for validation. Below are some Applications and Meta information are classified into there respective domains.

   party*=IM.DOCUMENT.IMAGE:IM.DOCUMENT.UPLOAD

   meta=GETMetadata:T24FieldMetadata

   Note: (*) asterisks indicates that domain is default to any Application that is not classified or categorized into its respective domain, in above example "party" is default domain. The application lists are separated by (:) semicolon. The file shall and must be located in the same directory as the batch files.

3. **verb.Properties**

   The file is used to map the IRIS Legacy command with corresponding IRISX methods. The methods are used to determine the type of the http operation and name of the operation as per the IRISX conventions. The file shall and must be located in the same directory as the batch files. Below mentioned are few entries from the verb properties file

   GetEntities=get

   GetEntity=get

   CreateEntity=create

   DeleteEntity=delete

   GET=get

   POST=create

   PUT=update

   DELETE=delete

4. vobcabulary-<domain>.json

   The file is used to catalog all domain specific entries like the domain, resource and property associated with a URL or exchange parameters keeping the entire exchange well known and secured. The vocabulary is part of IRISX conventions, any field(s) and criteria(s) exchange must meet the vocabulary. So owners can either update the vocabulary with artifact (VERSION\ENQUIRY) details or update the artifact based on the vocabulary. The recommended approach is to first check the vocabulary for the details and update the artifact if not meeting the expectation the vocabulary shall be updated in accordance. Below mentioned are few entries from domain specific vocabulary file indicating resource and property type resources.

```
{

"key" : "transactionType",

"description" : "transactionType",

"plural" : "transactionTypes",

"insteadOf" : [ ],

"links" : [ ],

"usage" : [ ],

"entryType" : "resource",

"dataType" : "String",

"label" : null,

"generated" : false

}, {

"key" : "transactionTotal",

"description" : "transactionTotal",

"plural" : "transactionTotals",

"insteadOf" : [ ],

"links" : [ ],

"usage" : [ ],

"entryType" : "property",

"dataType" : "String",

"label" : null,

"generated" : false

},
```

RIMImporter FAQ'S

1. Where to get the latest release pack of RimImporter from?

   Click Download to download the latest rimimporter release package from the Temenos Maven Repository

   Link : http://maven.temenosgroup.com/content/groups/temenos/com/temenos/irf/irf-rim-importer-packager/DEV.0.0-SNAPSHOT/

2. How to configure RimImporter for Remote T24Server?

   Refer : How to configure RimImporter to communicate Remote T24 Server?

Appendix

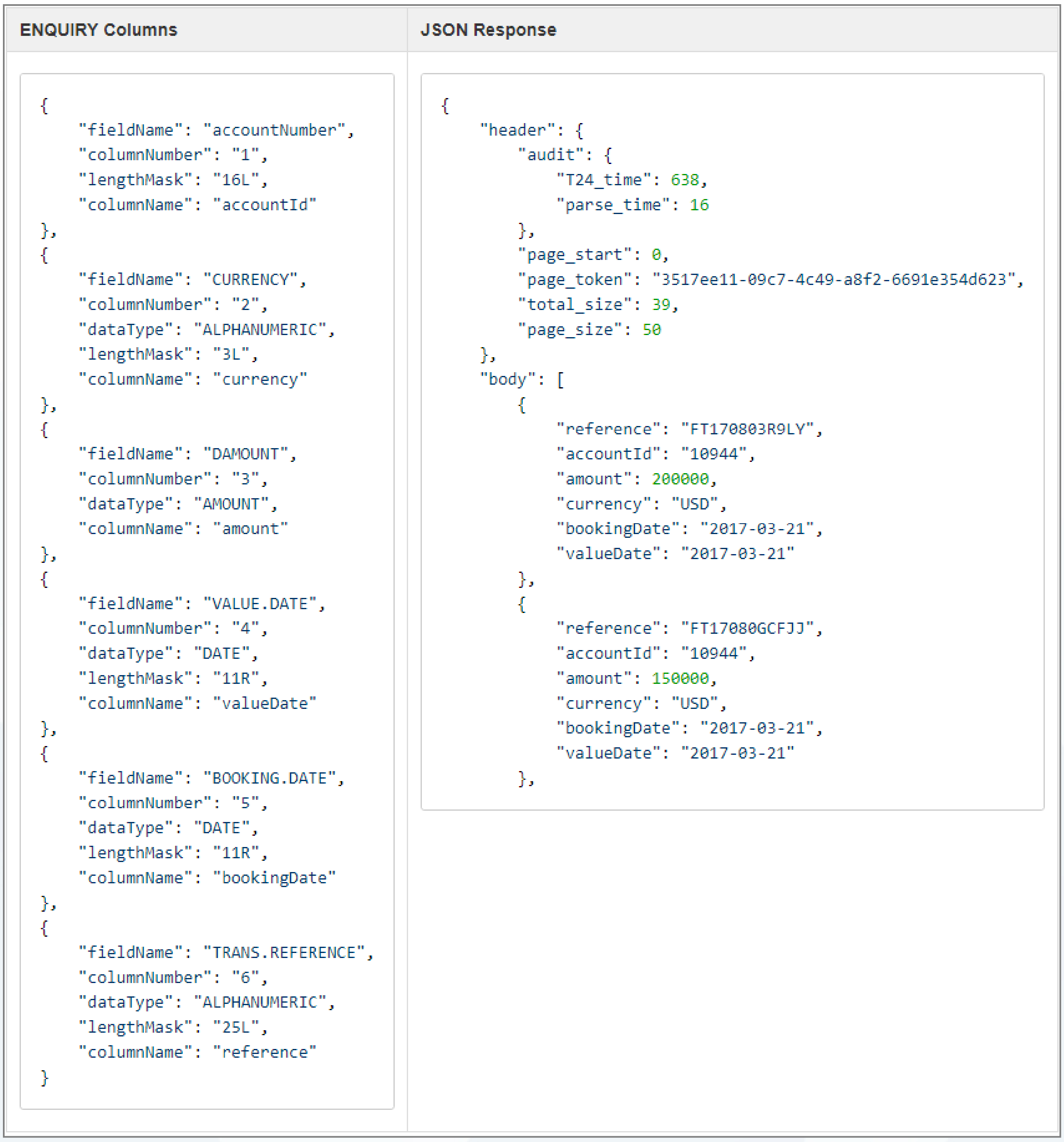
T24 Enquiry

Overview

This section explains how the T24 Enquiry configuration affects the structure of the JSON that is produced.

The Basics

A standard column based Enquiry is rendered as:



ENQUIRY Columns

```
{
    "fieldName": "accountNumber",
    "columnNumber": "1",
    "lengthMask": "16L",
    "columnName": "accountId"
},
{
    "fieldName": "CURRENCY",
    "columnNumber": "2",
    "dataType": "ALPHANUMERIC",
    "lengthMask": "3L",
    "columnName": "currency"
},
{
    "fieldName": "DAMOUNT",
    "columnNumber": "3",
    "dataType": "AMOUNT",
    "columnName": "amount"
},
{
    "fieldName": "VALUE.DATE",
    "columnNumber": "4",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "valueDate"
},
{
    "fieldName": "BOOKING.DATE",
    "columnNumber": "5",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "bookingDate"
},
{
    "fieldName": "TRANS.REFERENCE",
    "columnNumber": "6",
    "dataType": "ALPHANUMERIC",
    "lengthMask": "25L",
    "columnName": "reference"
}
```

JSON Response

```
{
    "header": {
        "audit": {
            "T24_time": 638,
            "parse_time": 16
        },
        "page_start": 0,
        "page_token": "3517ee11-09c7-4c49-a8f2-6691e354d623",
        "total_size": 39,
        "page_size": 50
    },
    "body": [
        {
            "reference": "FT170803R9LY",
            "accountId": "10944",
            "amount": 200000,
            "currency": "USD",
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        },
        {
            "reference": "FT17080GCFJJ",
            "accountId": "10944",
            "amount": 150000,
            "currency": "USD",
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        },
```

Adding Data to the Body

Setting the DISPLAY.BREAK to ONCE create a separate row in the response. Note the row count remains unchanged.

ENQUIRY Columns	JSON Response

ENQUIRY Columns

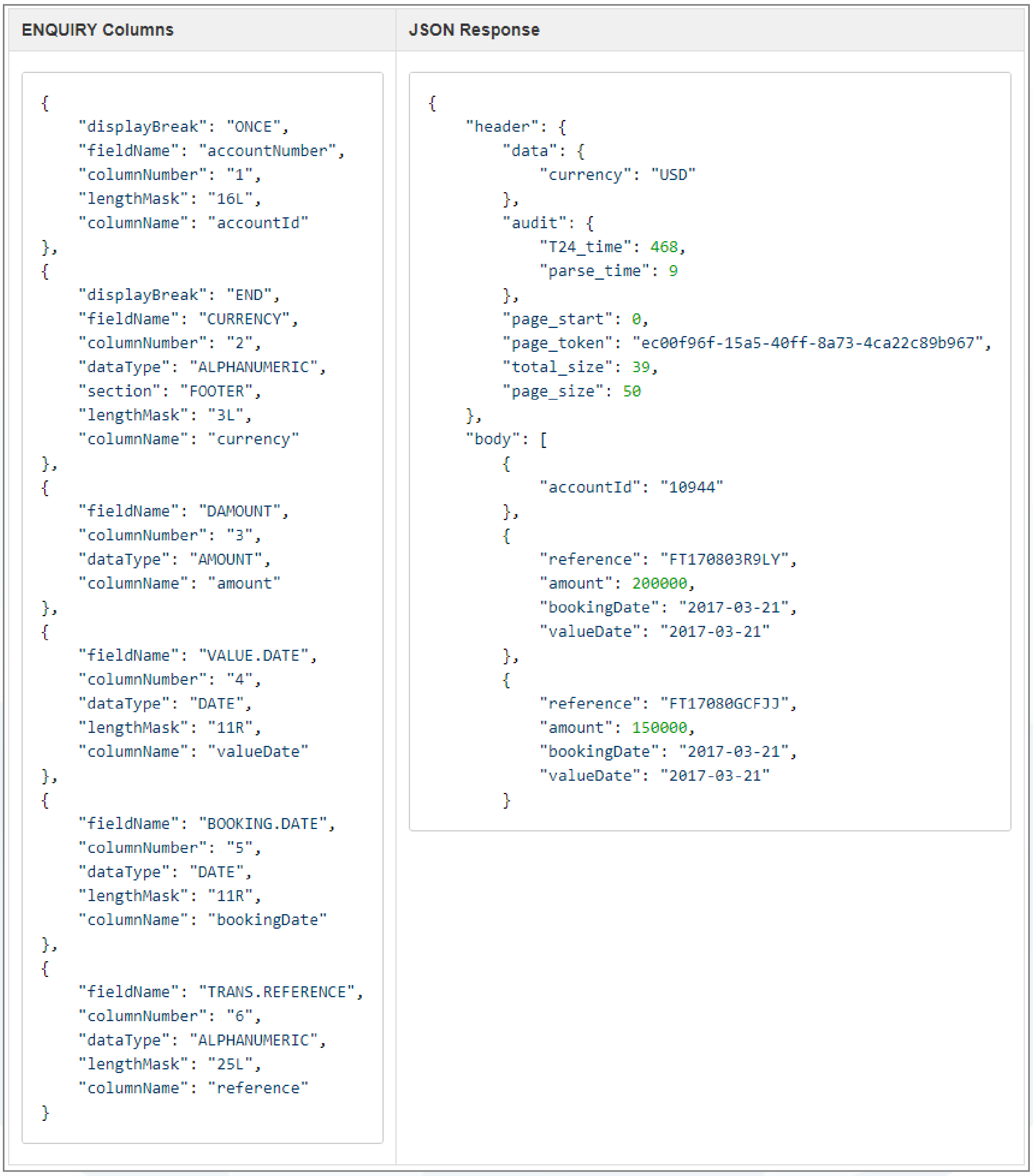
```
{
    "displayBreak": "ONCE",
    "fieldName": "accountNumber",
    "columnNumber": "1",
    "lengthMask": "16L",
    "columnName": "accountId"
},
{
    "fieldName": "CURRENCY",
    "columnNumber": "2",
    "dataType": "ALPHANUMERIC",
    "lengthMask": "3L",
    "columnName": "currency"
},
{
    "fieldName": "DAMOUNT",
    "columnNumber": "3",
    "dataType": "AMOUNT",
    "columnName": "amount"
},
{
    "fieldName": "VALUE.DATE",
    "columnNumber": "4",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "valueDate"
},
{
    "fieldName": "BOOKING.DATE",
    "columnNumber": "5",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "bookingDate"
},
{
    "fieldName": "TRANS.REFERENCE",
    "columnNumber": "6",
    "dataType": "ALPHANUMERIC",
    "lengthMask": "25L",
    "columnName": "reference"
}
```

JSON Response

```
{
    "header": {
        "audit": {
            "T24_time": 465,
            "parse_time": 7
        },
        "page_start": 0,
        "page_token": "9f8ea698-b52b-438e-817f-16e7552191c0",
        "total_size": 40,
        "page_size": 50
    },
    "body": [
        {
            "accountId": "10944"
        },
        {
            "reference": "FT170803R9LY",
            "amount": 200000,
            "currency": "USD",
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        },
        {
            "reference": "FT17080GCFJJ",
            "amount": 150000,
            "currency": "USD",
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        },
```

Adding Data to the Header

Setting the DISPLAY.BREAK to END and SECTION to FOOTER creates a data element in the header section of the response. Note the row count remains unchanged.



ENQUIRY Columns	JSON Response

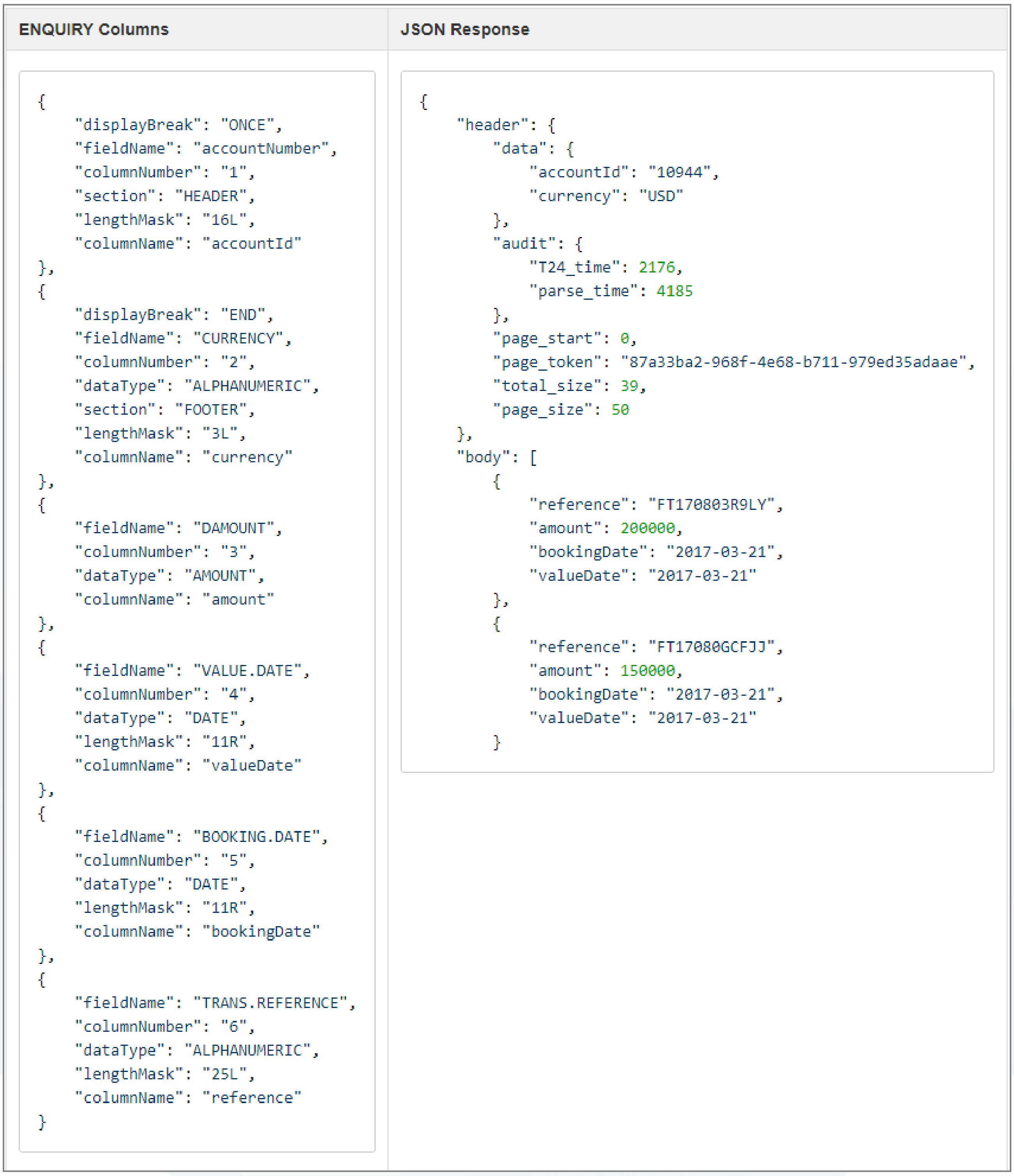
ENQUIRY Columns

```
{
    "displayBreak": "ONCE",
    "fieldName": "accountNumber",
    "columnNumber": "1",
    "lengthMask": "16L",
    "columnName": "accountId"
},
{
    "displayBreak": "END",
    "fieldName": "CURRENCY",
    "columnNumber": "2",
    "dataType": "ALPHANUMERIC",
    "section": "FOOTER",
    "lengthMask": "3L",
    "columnName": "currency"
},
{
    "fieldName": "DAMOUNT",
    "columnNumber": "3",
    "dataType": "AMOUNT",
    "columnName": "amount"
},
{
    "fieldName": "VALUE.DATE",
    "columnNumber": "4",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "valueDate"
},
{
    "fieldName": "BOOKING.DATE",
    "columnNumber": "5",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "bookingDate"
},
{
    "fieldName": "TRANS.REFERENCE",
    "columnNumber": "6",
    "dataType": "ALPHANUMERIC",
    "lengthMask": "25L",
    "columnName": "reference"
}
```

JSON Response

```
{
    "header": {
        "data": {
            "currency": "USD"
        },
        "audit": {
            "T24_time": 468,
            "parse_time": 9
        },
        "page_start": 0,
        "page_token": "ec00f96f-15a5-40ff-8a73-4ca22c89b967",
        "total_size": 39,
        "page_size": 50
    },
    "body": [
        {
            "accountId": "10944"
        },
        {
            "reference": "FT170803R9LY",
            "amount": 200000,
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        },
        {
            "reference": "FT17080GCFJJ",
            "amount": 150000,
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        }
```

Changing the SECTION to HEADER for accountId removes the added row, and moves the data to the header section. NB if multiple data elements are present, i.e. no DISPLAY.BREAK is set, the last element will be added to the header. Note the row count remains unchanged.



ENQUIRY Columns

```
{
    "displayBreak": "ONCE",
    "fieldName": "accountNumber",
    "columnNumber": "1",
    "section": "HEADER",
    "lengthMask": "16L",
    "columnName": "accountId"
},
{
    "displayBreak": "END",
    "fieldName": "CURRENCY",
    "columnNumber": "2",
    "dataType": "ALPHANUMERIC",
    "section": "FOOTER",
    "lengthMask": "3L",
    "columnName": "currency"
},
{
    "fieldName": "DAMOUNT",
    "columnNumber": "3",
    "dataType": "AMOUNT",
    "columnName": "amount"
},
{
    "fieldName": "VALUE.DATE",
    "columnNumber": "4",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "valueDate"
},
{
    "fieldName": "BOOKING.DATE",
    "columnNumber": "5",
    "dataType": "DATE",
    "lengthMask": "11R",
    "columnName": "bookingDate"
},
{
    "fieldName": "TRANS.REFERENCE",
    "columnNumber": "6",
    "dataType": "ALPHANUMERIC",
    "lengthMask": "25L",
    "columnName": "reference"
}
```

JSON Response

```
{
    "header": {
        "data": {
            "accountId": "10944",
            "currency": "USD"
        },
        "audit": {
            "T24_time": 2176,
            "parse_time": 4185
        },
        "page_start": 0,
        "page_token": "87a33ba2-968f-4e68-b711-979ed35adaae",
        "total_size": 39,
        "page_size": 50
    },
    "body": [
        {
            "reference": "FT170803R9LY",
            "amount": 200000,
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        },
        {
            "reference": "FT17080GCFJJ",
            "amount": 150000,
            "bookingDate": "2017-03-21",
            "valueDate": "2017-03-21"
        }
```

Multi-Values as Collections

To create a multi value collection group, the **COMMENTS** field is used to define the collection label, e.g. currencies. **SINGLE.MULTI** must be set to S and remove any width restrictions in **LENGTH.MASK**. By default, IRIS will separate multi values base on a SPACE character (the T24 default) - however some ENQUIRYs are defined to delimit by a pipe character (|). If this is the case, the **COMMENTS** field is also used to specifiy the delimieter, hence a multi-value label of currencies used with a pipe delimiter would be specified as currencies PIPE.

T24 Version

Overview

This section explains how the T24 Version configuration affects the structure of the JSON that is produced.

The Basics

Version Field	Usage in API	Description
DESCRIPTION	Operation	Used as the name of the operation in the API, should conform to the following pattern: verbCollectionResource e.g. GetAccountTransactions CreateAccountTransfer
TEXT	Label to be used in JSON	The label to be used in the JSON response **Note**: • The text label provided must match with the entries in the vocabulary • Add new text label in IRIS R18 Vocabulary if it doesn't exists else reuse closest matching entry as text label

PROMPT.TEX-T	Group labels	Used to signify that a field is part of a collection (multi-value group) or collection of collections (sub-value group) and to provide the name of the collection.
		
		For example, the fields chargeFor, chargeType and chargeAmount are a multi value set. Setting the PROMPT.TEXT to **charges** would result in the following JSON strcuture:
		
		`"charges": [ { "chargeFor": "string", "chargeType": "string", "chargeAmount": "string" } ]`
		
		Where a sub-value is set, the group names are needed for both the multi value group and the sub value group, separated by a space.
		
		For example, a multi-value set of relatedMsg has a sub-value of timeIndicator. Setting the PROMPT.TEXT to relatedMessages for the relatedMsg field and to relatedMessages timeIndicators for timeIndicator, would result in the following JSON strcuture:
		
		`"relatedMessages": [ { "timeIndicators": [ { "timeIndicator": "string" } ], "relatedMsg": "string" } ]`
		
		NB If PROMPT.TEXT is not set, the fields are NOT noted as a collection

ATTRIBS	Data type of the field.	Alphanumeric Date Amount
TOOL.TIP	List of allowed values	Comma delimited list of allowed values