DAVID ARNOLD

daverno@au1.ibm.com

Abstract

Leveraging the ACE micro services 1,2 and 3 on RHOS and testing them with IBM (Rational) Integration Tester. The IBM (Rational) Service Virtualization Tester will also be used for mocking and stubbing ACE MS2 and an MQ backend application

IBM Middleware Tested on RHOS with IBM (Rational) Integration Tester

Testing ICP4i (ACE and MQ) custom images on RHOS 4.2 with Rational Integration Tester

Contents

[Using IBM (Rational) Integration Tester with ACE custom images on OpenShift 3](#_Toc36646855)

[Introduction 3](#_Toc36646856)

[Assuming the role of the integration developer we will explore using components of the IBM (Rational) Test Workbench for: 3](#_Toc36646857)

[Testing Environment 3](#_Toc36646858)

[Overview Diagram 4](#_Toc36646859)

[IBM (Rational) Integration Tester – downloads 4](#_Toc36646860)

[IBM Integration Tester 4](#_Toc36646861)

[IBM installation manager for the IBM Integration Tester install 5](#_Toc36646862)

[Download/clone the IBM (Rational) Integration Tester project 5](#_Toc36646863)

[Open IBM Integration Tester and direct to workspace 5](#_Toc36646864)

[Navigating the IBM (Rational) Integration Tester 7](#_Toc36646865)

[Architecture School - Schema library 7](#_Toc36646866)

[Architecture School - Physical View 7](#_Toc36646867)

[Architecture School - Logical View 9](#_Toc36646868)

[Test Factory – Tests 11](#_Toc36646869)

[Test Factory – Stubs 13](#_Toc36646870)

[Explore the Test Lab 14](#_Toc36646871)

[1.Run and test ACE “Stubs” and ACE Integration Services 17](#_Toc36646872)

[Mocking ACE Microservice 2 with an IBM Test Virtualization Server stub 17](#_Toc36646873)

[Run the ACE Micro Service 2 stub 17](#_Toc36646874)

[Test ACE Micro Service 2 stub with a local REST client 18](#_Toc36646875)

[Testing the ACE Microservice 2 stub using IBM Integration Tester locally and via a 23](#_Toc36646876)

[Public IP address 23](#_Toc36646877)

[IBM Secure Gateway Service – Server side 23](#_Toc36646878)

[IBM Secure Gateway Client side 24](#_Toc36646879)

[Windows firewalls 27](#_Toc36646880)

[Test calling the RTVS ACE MS2 stub via public IP 28](#_Toc36646881)

[Use IBM Integration Tester to test ACE Microservice 1 (on RH Openshift) connected to ACE Microservice 2 Stub 31](#_Toc36646882)

[Get the ACE MS1 route details from RH OpenShift console 32](#_Toc36646883)

[Start/Restart the IBM Rational Integration Tester ACE MS2 Stub 33](#_Toc36646884)

[Use a REST client to call ACE MS1 (which in turn called ACE MS2 stub) 34](#_Toc36646885)

[Observe results in ACE MS 2 stub console 35](#_Toc36646886)

[Invoking ACE MS1 on RH OpenShift from a IBM Rational Integration Tester 36](#_Toc36646887)

[Observe the results in the console for the ACE MS2 stub 38](#_Toc36646888)

[Observe results in the console for the testservice 1 38](#_Toc36646889)

[2.Run and test ACE microservices deployed on RH OpenShift that connect to MQ 41](#_Toc36646890)

[Using ACE Micro Service 3 (MS3) with a remote MQ Queue Manager 41](#_Toc36646891)

[Configure and test an MQ Queue Manage on your local IBM Integration Tester 41](#_Toc36646892)

[MQ SVRCONN Channel creation for Linux based queue manager 41](#_Toc36646893)

[MQ SVRCONN Channel creation for windows based queue manager 42](#_Toc36646894)

[Set up IBM Secure Gateway Service for Public IP to Queue Manager MyAPPQMGR 43](#_Toc36646895)

[IBM Cloud Secure Gateway - Server side 43](#_Toc36646896)

[IBM Secure Gateway Service – Client (Laptop end) 46](#_Toc36646897)

[Windows firewall 46](#_Toc36646898)

[Using/Matching Secure Gateway parameters on ACE MS 3 MQOutput configuration 47](#_Toc36646899)

[Testing MQ via IBM Public IP – using RFHUTILC (MQ client test) 50](#_Toc36646900)

[Check the result on the target queue manager 51](#_Toc36646901)

[Testing ACE Microservice 3 to Put to MQ via IBM Public IP – Using a REST Client 52](#_Toc36646902)

[Obtain the RH OpenShift Route for ACE MS3 52](#_Toc36646903)

[URL and test data for calling ACE MS3 service 52](#_Toc36646904)

[Test with REST Client 53](#_Toc36646905)

[Check Test Results on MQ 54](#_Toc36646906)

[Using the MQ Stub on IBM Rational Integration Tester as a backend application 55](#_Toc36646907)

[Review stub parameters in IBM Rational Integration Tester 55](#_Toc36646908)

[Stub uses IVT.SVRCONN channel definition in the MyAPPQMGR 58](#_Toc36646909)

[Start IBM Rational Integration Tester Stub to retrieve messages placed on MQ by ACE MS3 59](#_Toc36646910)

[IBM Rational Integration Tester Stub receives message from ACE MS3 60](#_Toc36646911)

[Review MyAPPQMGR queue FROM.BACKEND.Q contents 60](#_Toc36646912)

# Using IBM (Rational) Integration Tester with ACE custom images on OpenShift

## Introduction

## Assuming the role of the integration developer we will explore using components of the IBM (Rational) Test Workbench for:

* Mocking (Stubbing) and Testing ACE integration microservices with IBM (Rational) Integration Tester
* Mocking (Stubbing) and Testing ACE with MQ integration microservices with IBM (Rational) Integration Tester

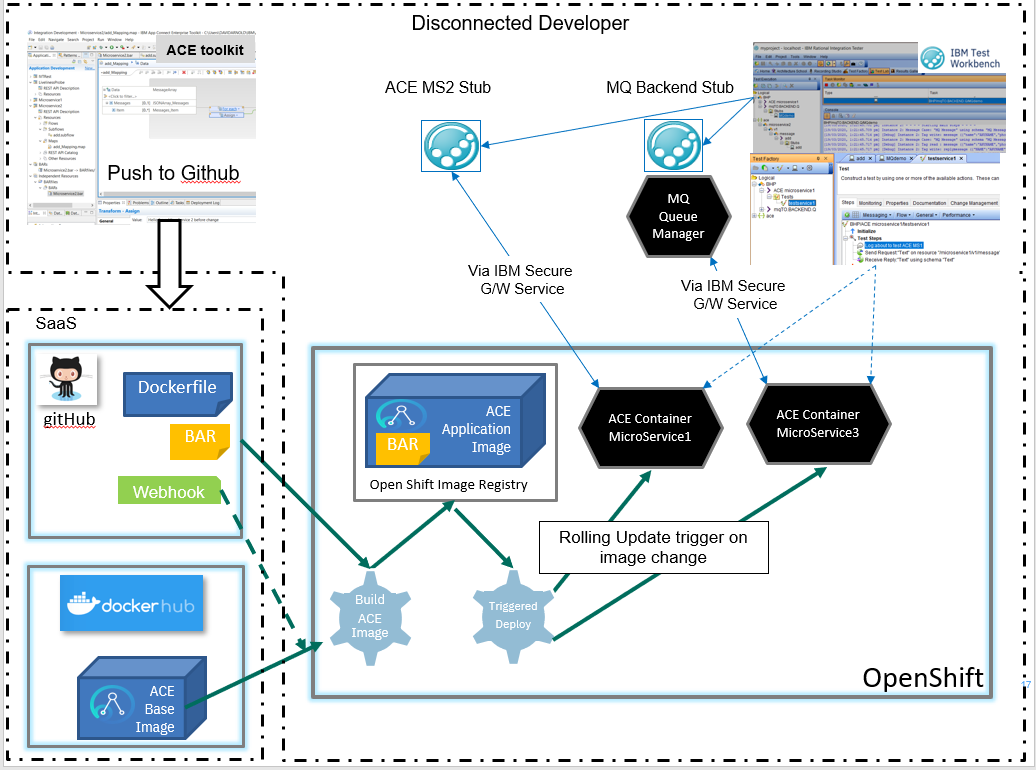
1. Run and test ACE “Stubs” and ACE Integration Services
   1. Mocking ACE Microservice 2 with an IBM Test Virtualization Server stub
   2. Testing the ACE Microservice 2 stub locally and via a Public IP address
   3. Use IBM Integration Tester to test ACE Microservice 1 (on RH Openshift) connected to ACE Microservice 2 Stub.
2. Run and test ACE microservices deployed on RH OpenShift that connect to MQ
   1. Configure and test an MQ Queue Manager on your local IBM Integration Tester machine/laptop
   2. Reconfigure ACE microservice 3 on RH Openshift to place a message on MQ for consumption by a “stub”
   3. Use IBM Integration Tester to test ACE Microservice 3 (on RH Openshift) connected to MQ
   4. Use IBM Test Virtualization Server MQ backend stub to Get and Put messages

## Testing Environment

In the examples on this document.

* IBM ACE Toolkit is on a Windows T480 laptop
* IBM (Rational) Integration Tester (with Virtualization Test Server) on Windows T480 laptop
* RH OpenShift 4.2 on IBM Public Cloud
* IBM Cloud Secure Gateway Service is used to offer public IP addresses to the T480 laptop
  + Note: CodeReady, Minishift or similar could have be used locally on the laptop
  + The Secure Gateway service would not have been required in that case

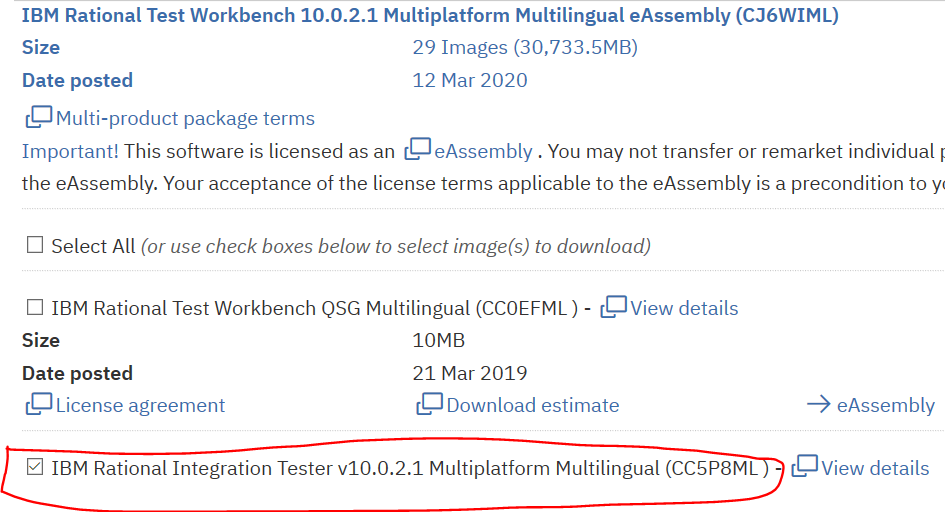
## Overview Diagram



## IBM (Rational) Integration Tester – downloads

Download IBM Rational Integration Tester v10.0.2.1. It is part of the IBM Rational Test Workbench v10.0.2.1.

### IBM Integration Tester



### IBM installation manager for the IBM Integration Tester install

Download and install the IBM Installation manager from the following site:

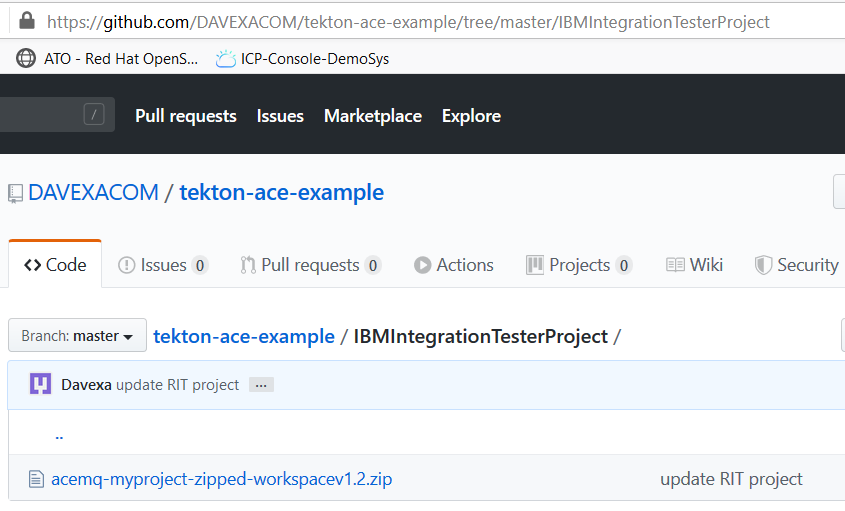
<https://jazz.net/downloads/ibm-installation-manager/releases/1.9.1.1/agent.installer.win32.win32.x86_64_1.9.1001.20191112_1525.zip>

Use the installation manager to install IBM (Rational) Integration Tester

## Download/clone the IBM (Rational) Integration Tester project

The collateral for these tutorials is available from github - https://github.com/DAVEXACOM/tekton-ace-example

<https://github.com/DAVEXACOM/tekton-ace-example/tree/master/IBMIntegrationTesterProject>

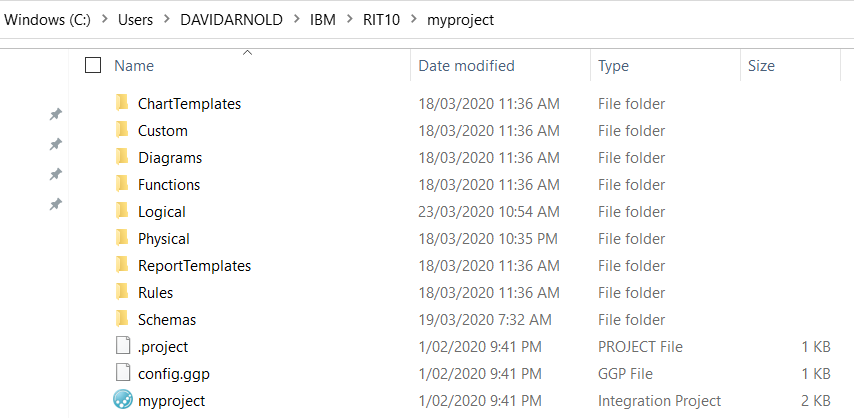


The acemq-myproject-zipped-workspacev1.2.zip is a ZIP of the IBM Integration Tester workspace for testing and stubbing the ACE and MQ artefacts.

### Open IBM Integration Tester and direct to workspace

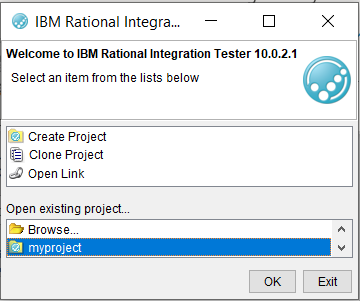
Unzip the IBM Integration Tester workspace acemq-myproject-zipped-workspacev1.2.zip to your local file system. For example:

c:\users\username\IBM\RIT10



Open the IBM (Rational) Integration Tester.

The first time the tool is opened you will Browse to open an existing project called myproject. Subsequently, you can simply open an existing project from the pull down.



## Navigating the IBM (Rational) Integration Tester

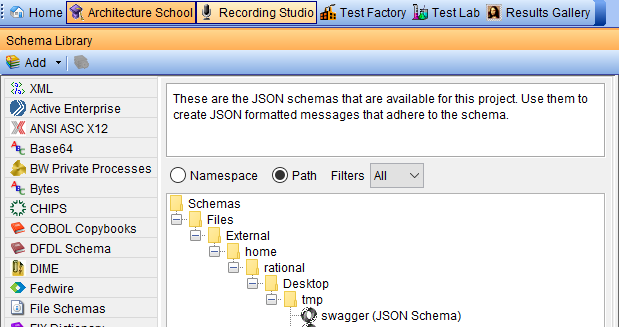
In this section you will simply explore the different views and tabs in the IBM (Rational) Integration Tester such that your get a “feel” for the tooling.

### Architecture School - Schema library

Description: In the Schema Library, you can view and manage the schemas and message formats that are available in your project.

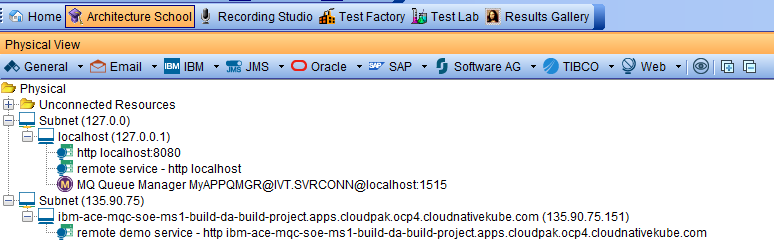
Schema library is where you can manage the schemas used for protocols you are using in this projects. Schema is important, and makes your testing and stub development easy.

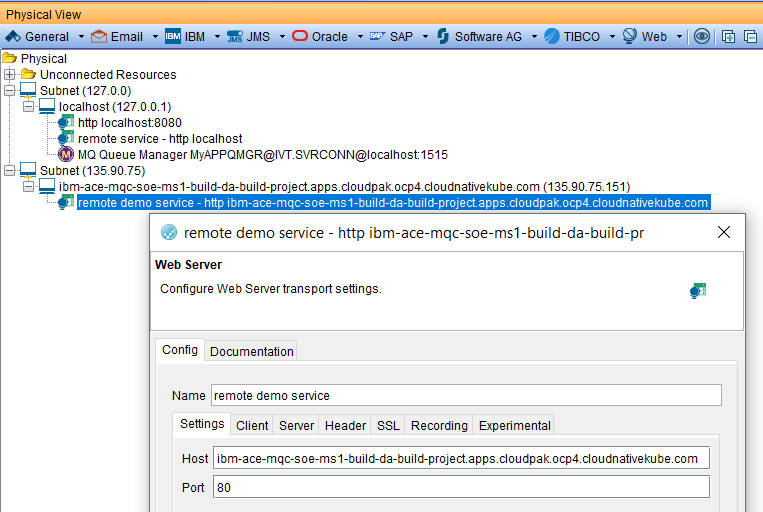
ACE Microservice open api documents imported here

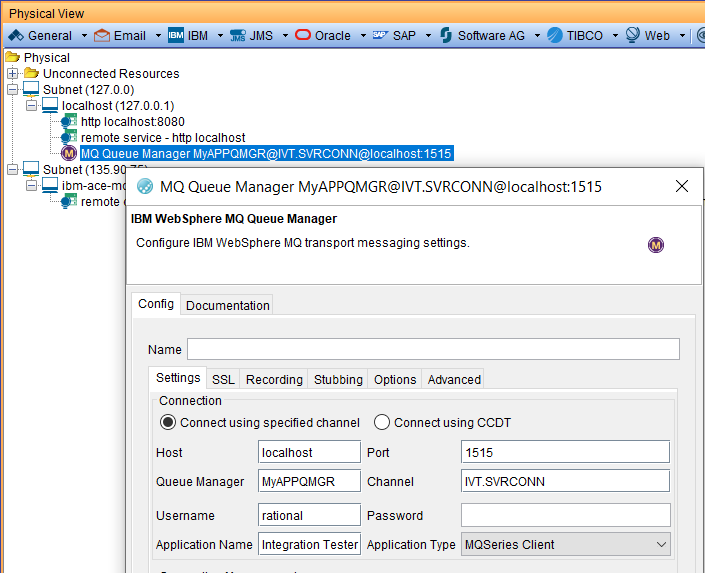


### Architecture School - Physical View

Description: The Physical View displays available physical resources and their location within the enterprise. Resources in this view are organized by subnet and host. If a resource is not associated with a subnet or host, it is displayed under Unconnected Resources.



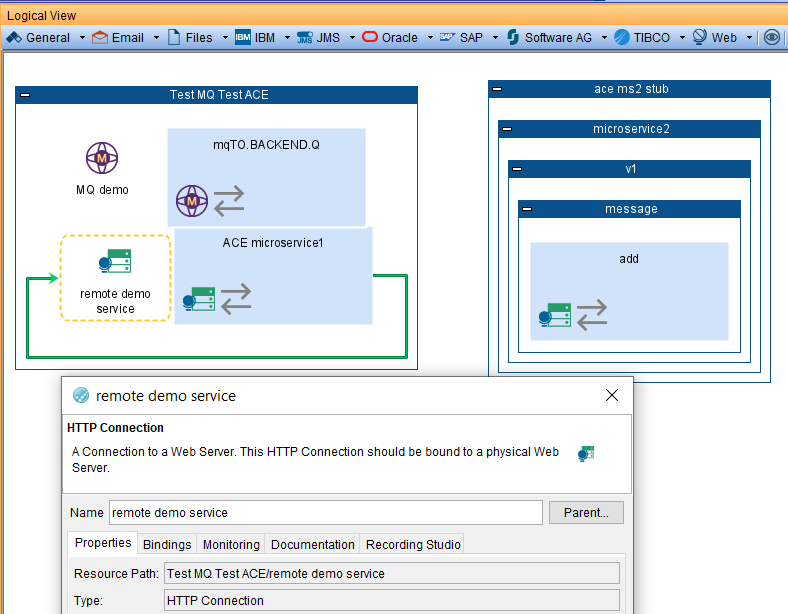




### Architecture School - Logical View

Description: The Logical View provides an abstract model of the system under test. You can visualize the Service and Infrastructure Components, their dependencies, inputs and outputs, and the interface patterns in this view.

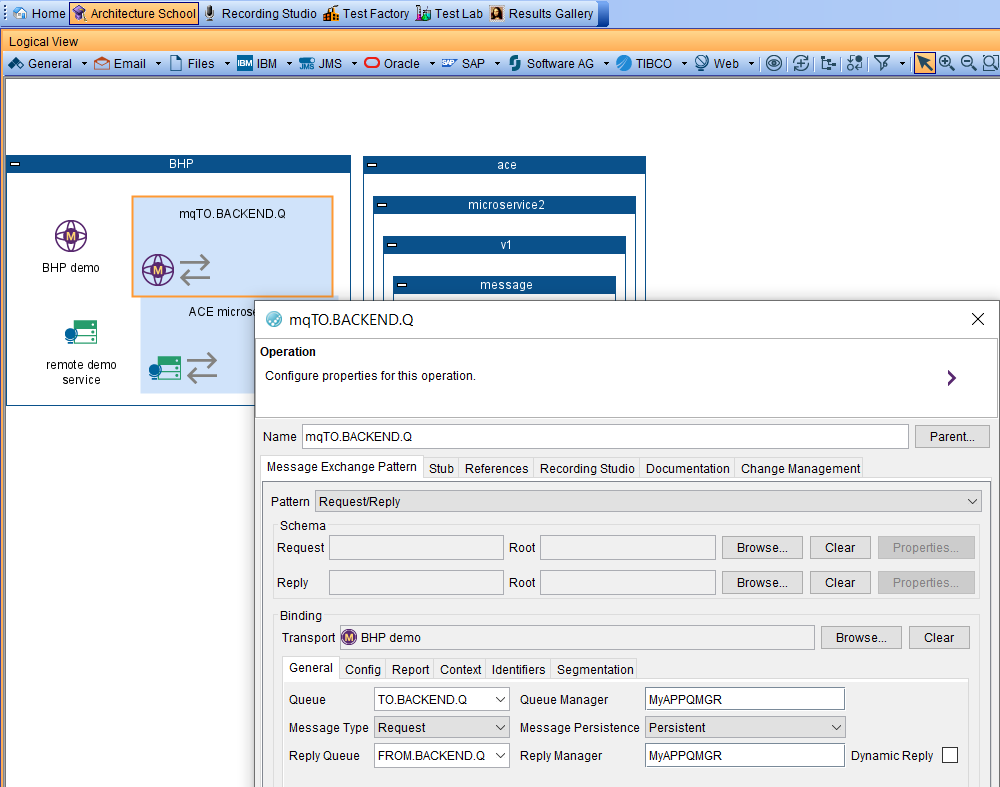
Logical Resource are the resources used in test/stub, rather than the Physical Resource defined in Physical View. You can bind the logical and physical resource and link them together in this view.



In “Bindings” sub-tab, you can link physical and logical resources.

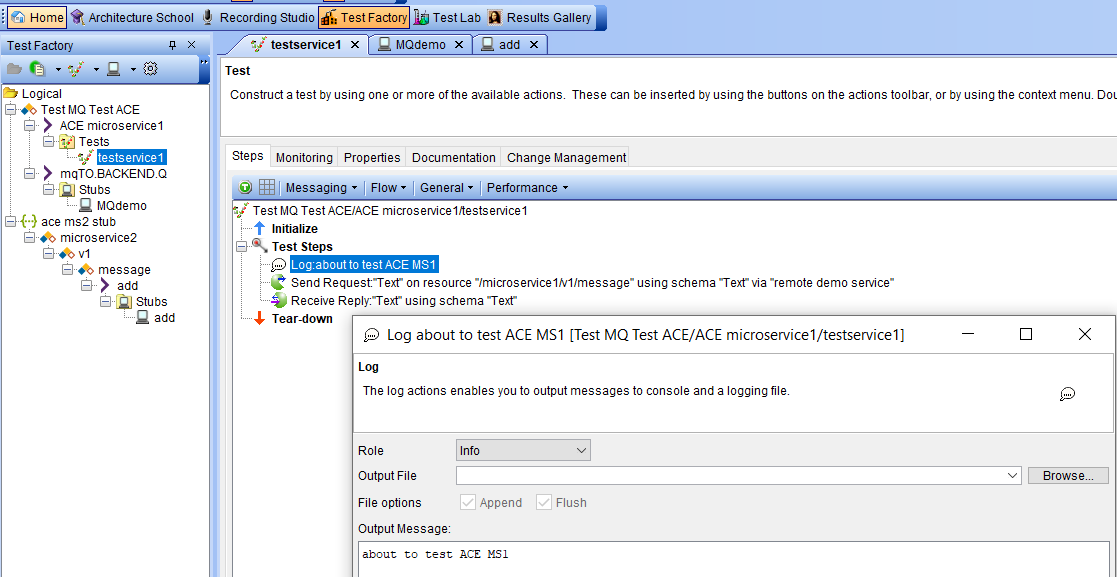


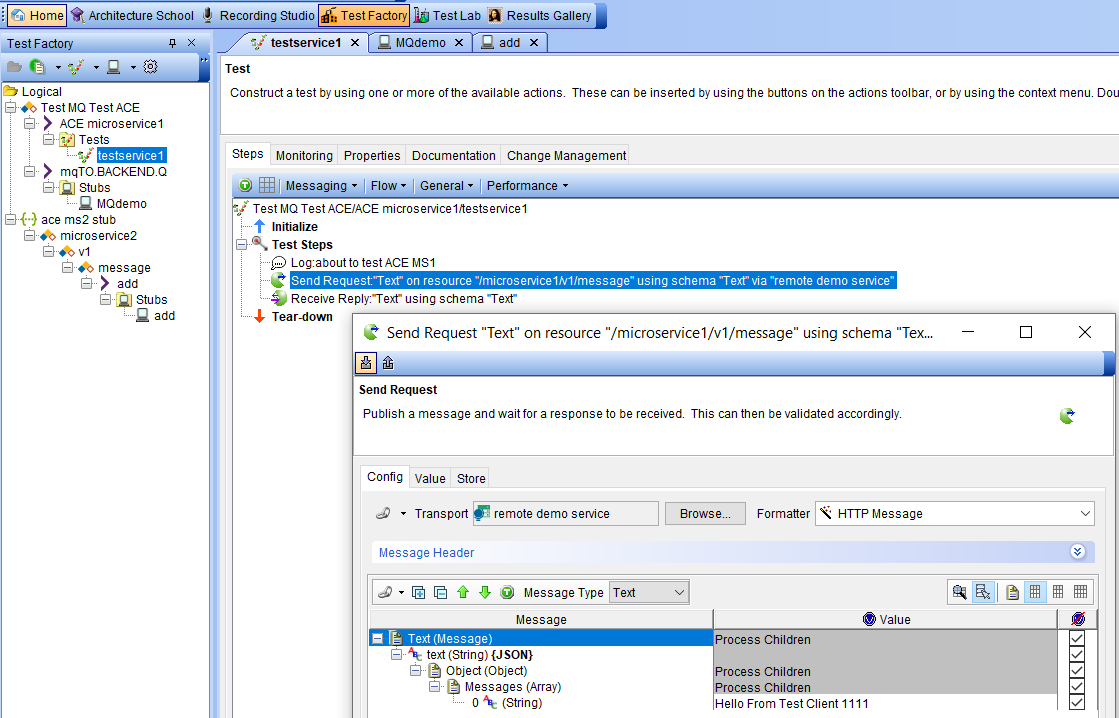
In possible, it’s better set the Schema for Request and Reply in this window.

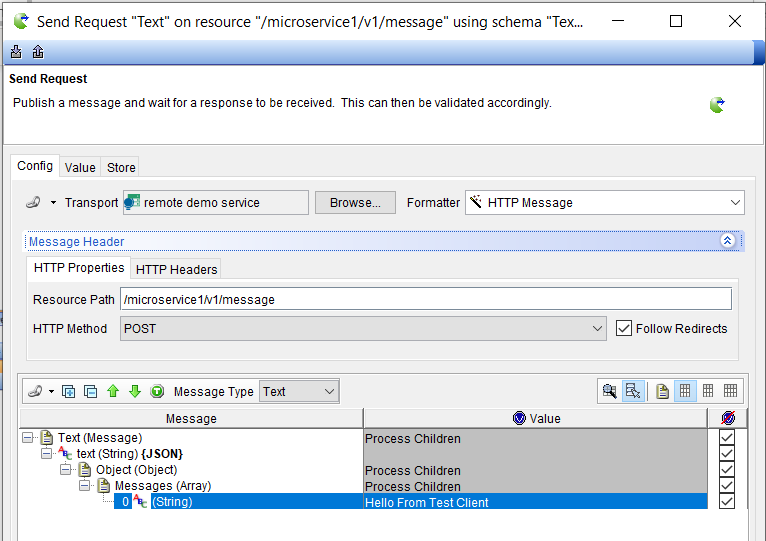


### Test Factory – Tests

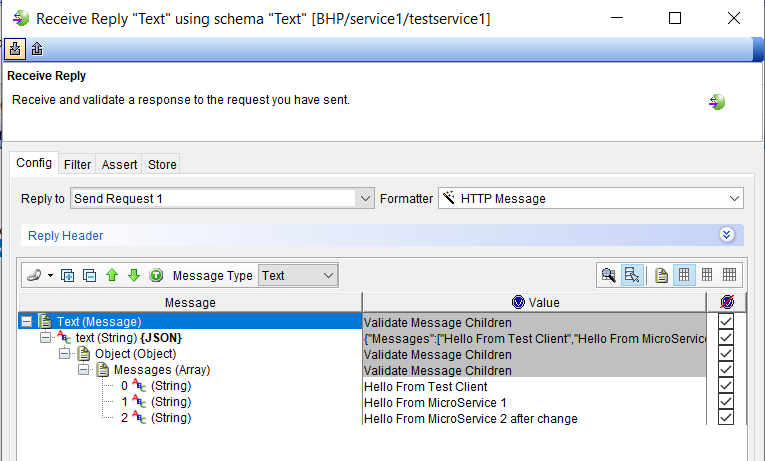
Description: The Test Factory perspective is where you develop the test cases and stubs.







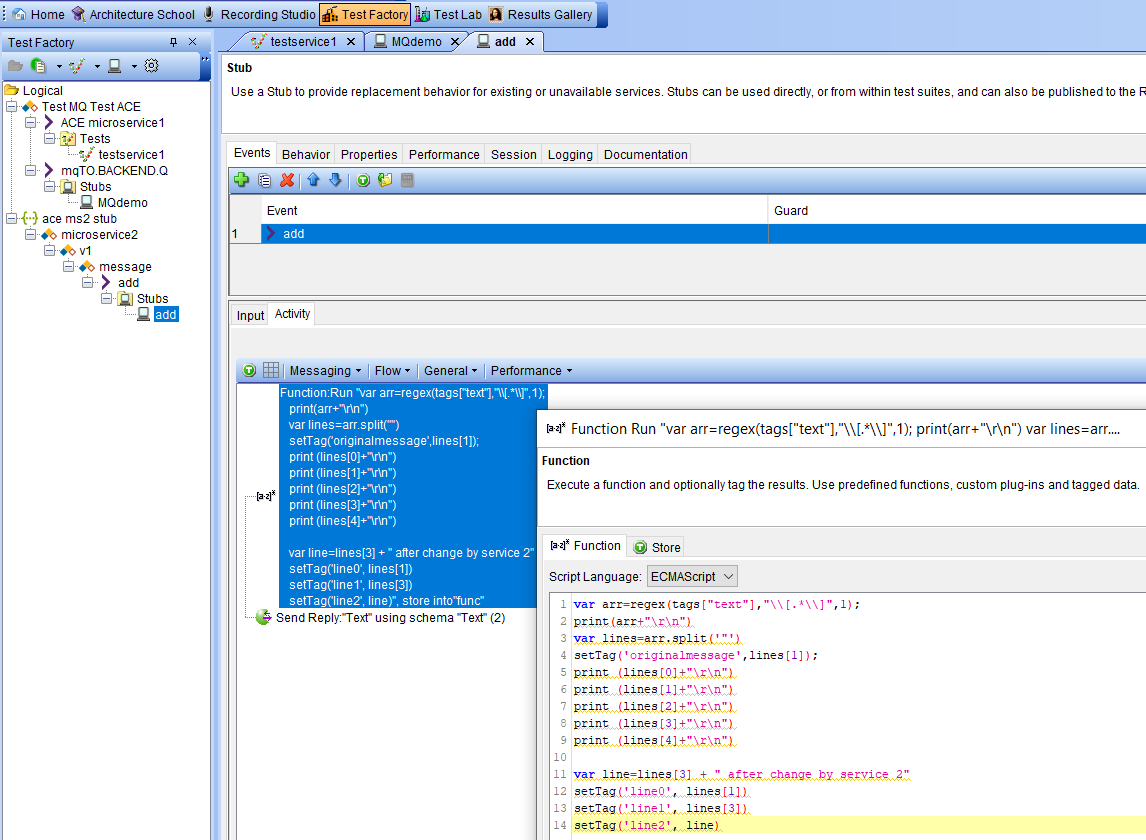
Defines the expected result – for comparison purposes

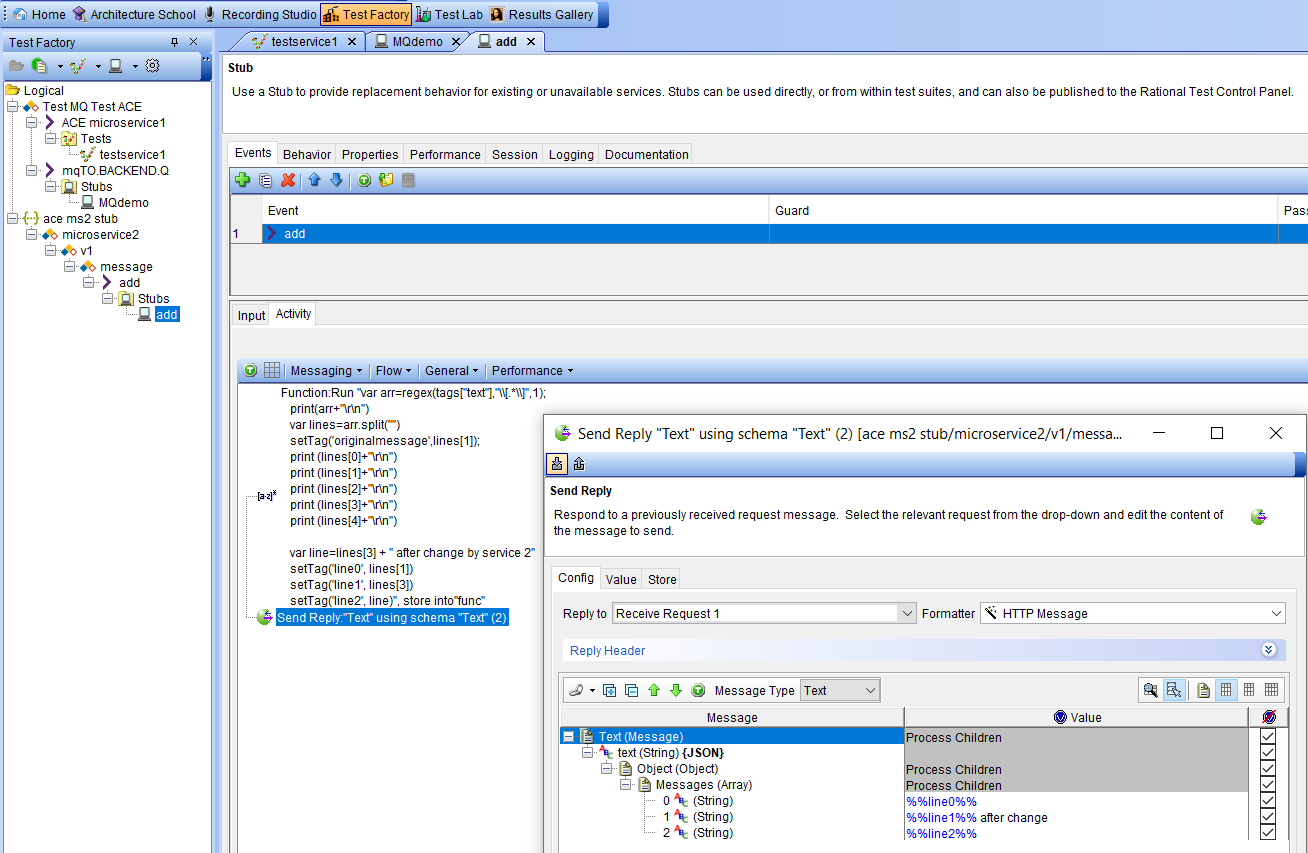


### Test Factory – Stubs

Description: The Test Factory perspective is where you develop the test cases and stubs.

#### Explore the ACE MS2 Stub

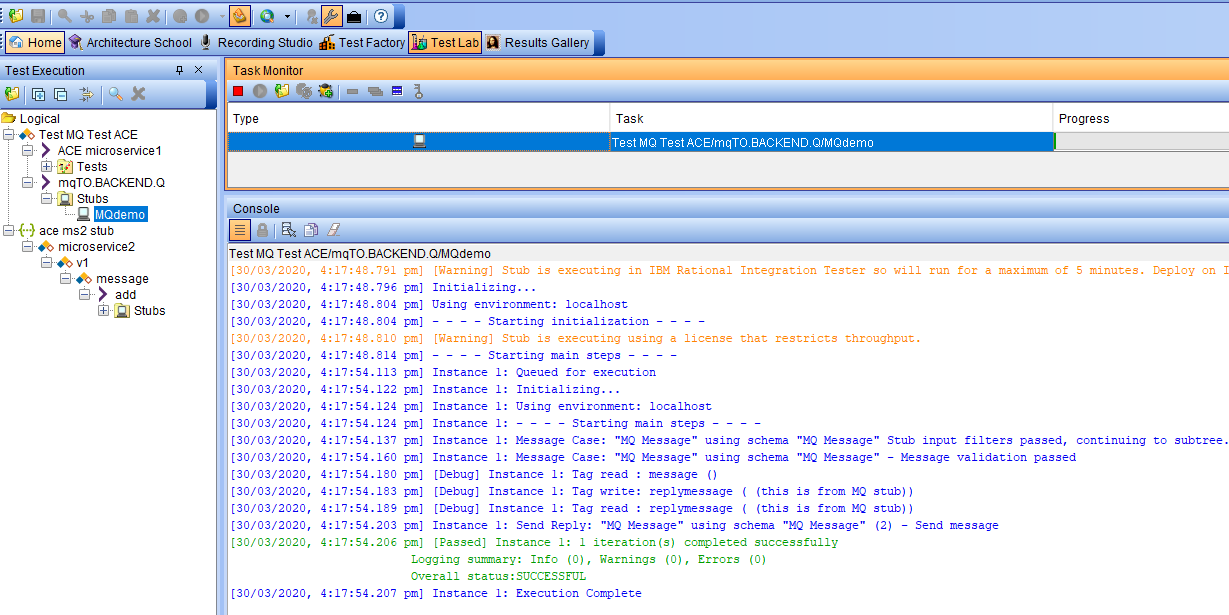




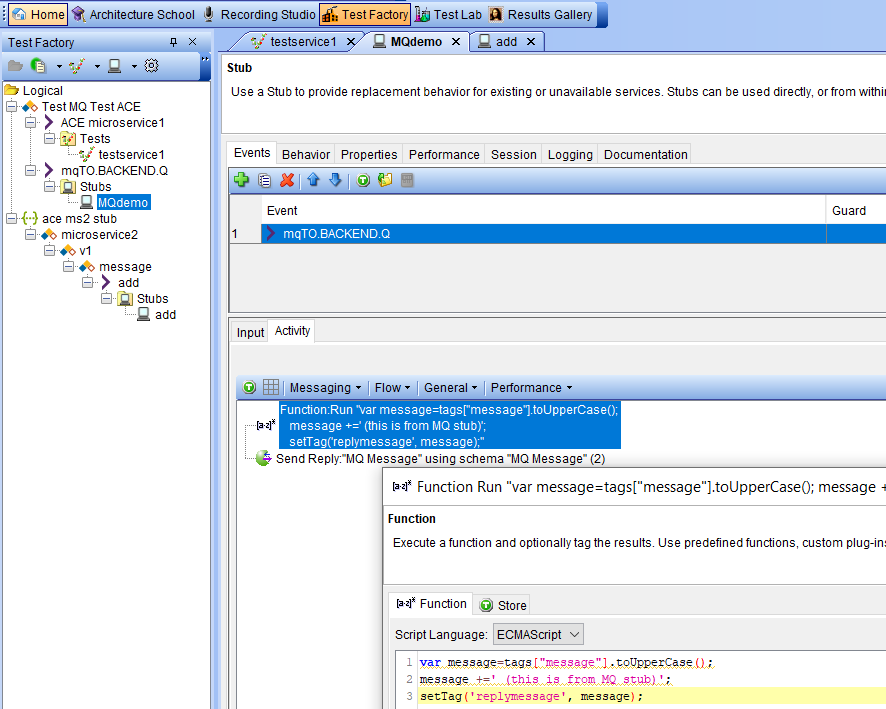
### Explore the Test Lab

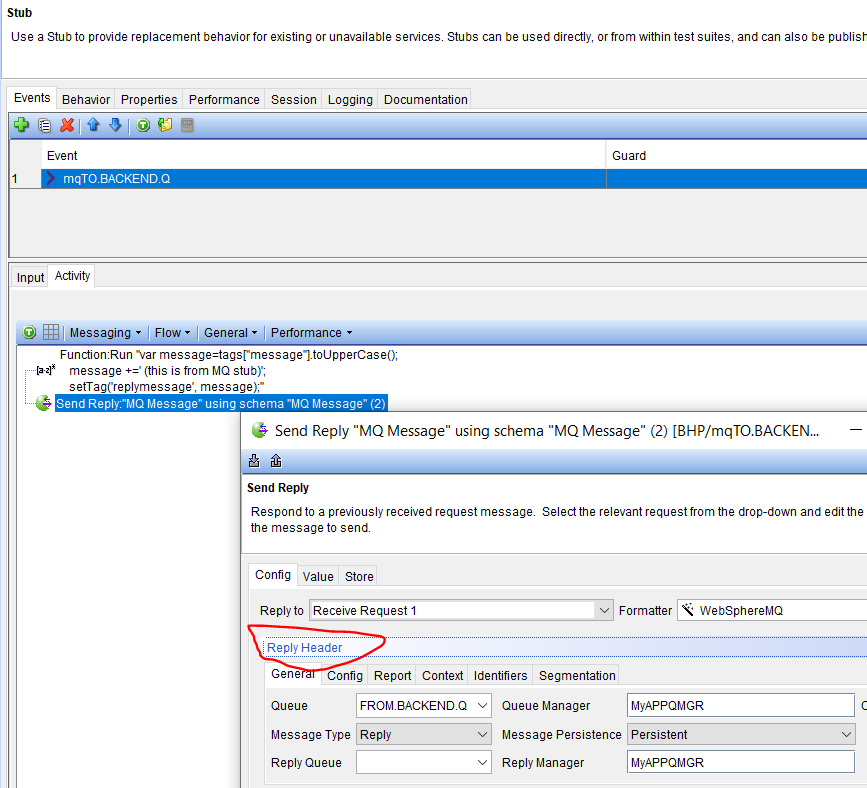
Description: Test Lab is where you execute test case and stubs. You can’t modify them here, you need to modify them in Test Factory.

In the Test Lab you can observer running tests and stubs



#### Explore the MQ Stub





Header can be expanded to use/set the content inside.

# 1.Run and test ACE “Stubs” and ACE Integration Services

## Mocking ACE Microservice 2 with an IBM Test Virtualization Server stub

We will first use a local rest client to test an ACE Micro Service 2 Stub on IBM (Rational) Integration Tester

### Run the ACE Micro Service 2 stub

In IBM Rational Integration Tester, stubs will only stay active for 5 minutes. So you will need to restart the stubs if the 5 minutes expires.

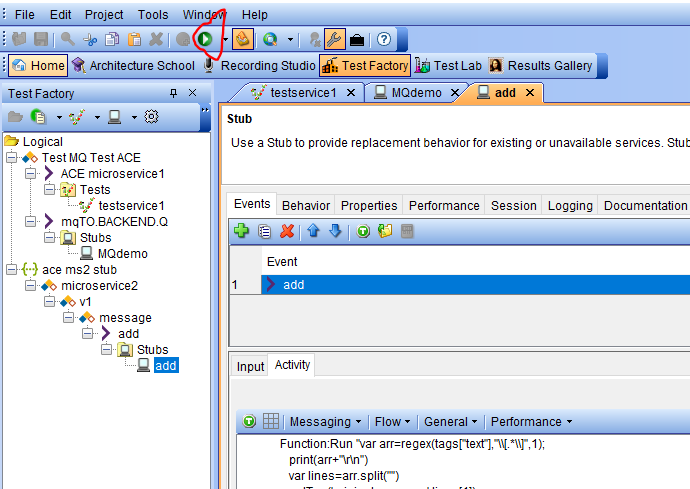
Note: In the full IBM Rational Test Virtualization Server, stubs can be configured to stay active indefinitely if required.

Select the Test Factory OR Test Lab tabs

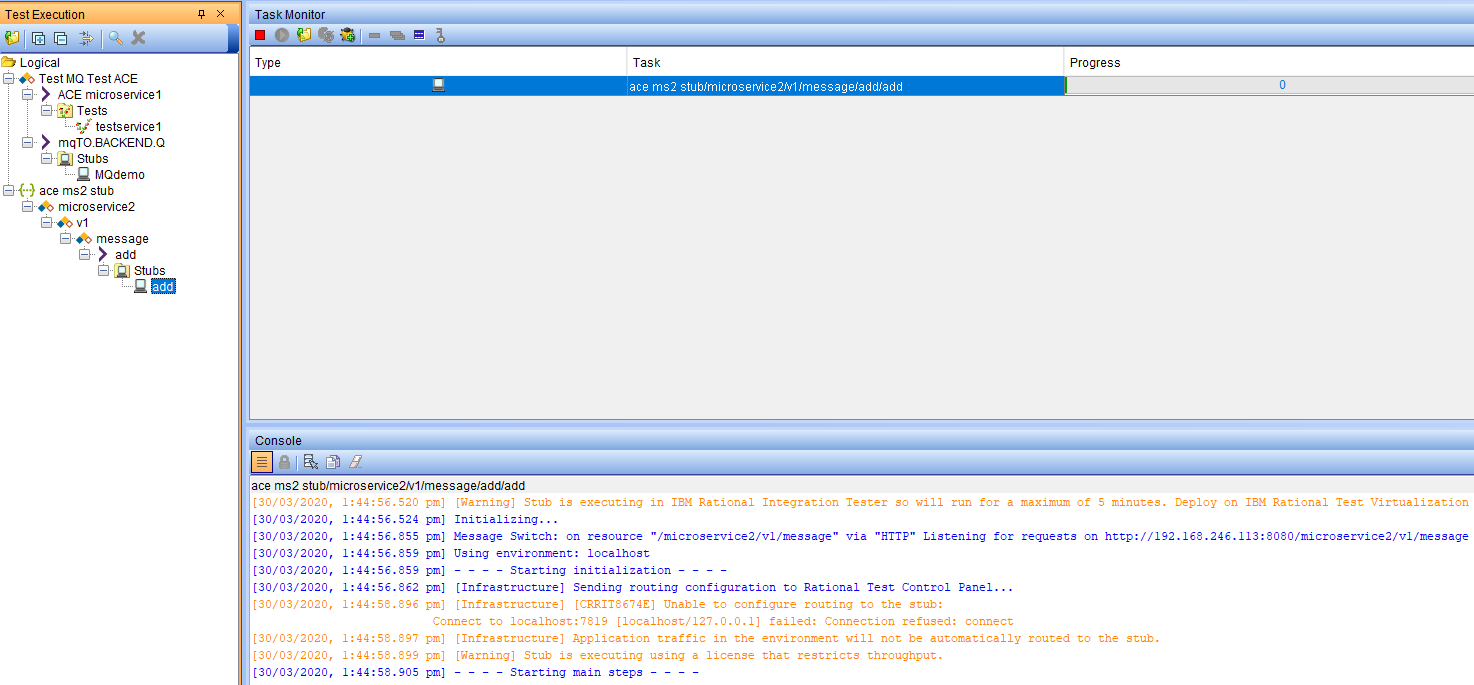
Expand the Logical file hierarchy

Ace ms2 stub->microservice2->v1->message->add->stubs->add

Double click on “add” then hit the circular green “Play” button to start the stub running



In the Task Monitor pane you will see the stub task running and in the console pane at the bottom of the screen you can see the stdout and stderr from the stub as it executes.



### Test ACE Micro Service 2 stub with a local REST client

We’ll use a REST client to test the ACE microservice 2 stub directly the first time.

POST to the following URL

http://localhost:8080/microservice2/v1/message

with data

{"Messages":["Hello From Test Client 1111"]}

You should get the following response returned

{

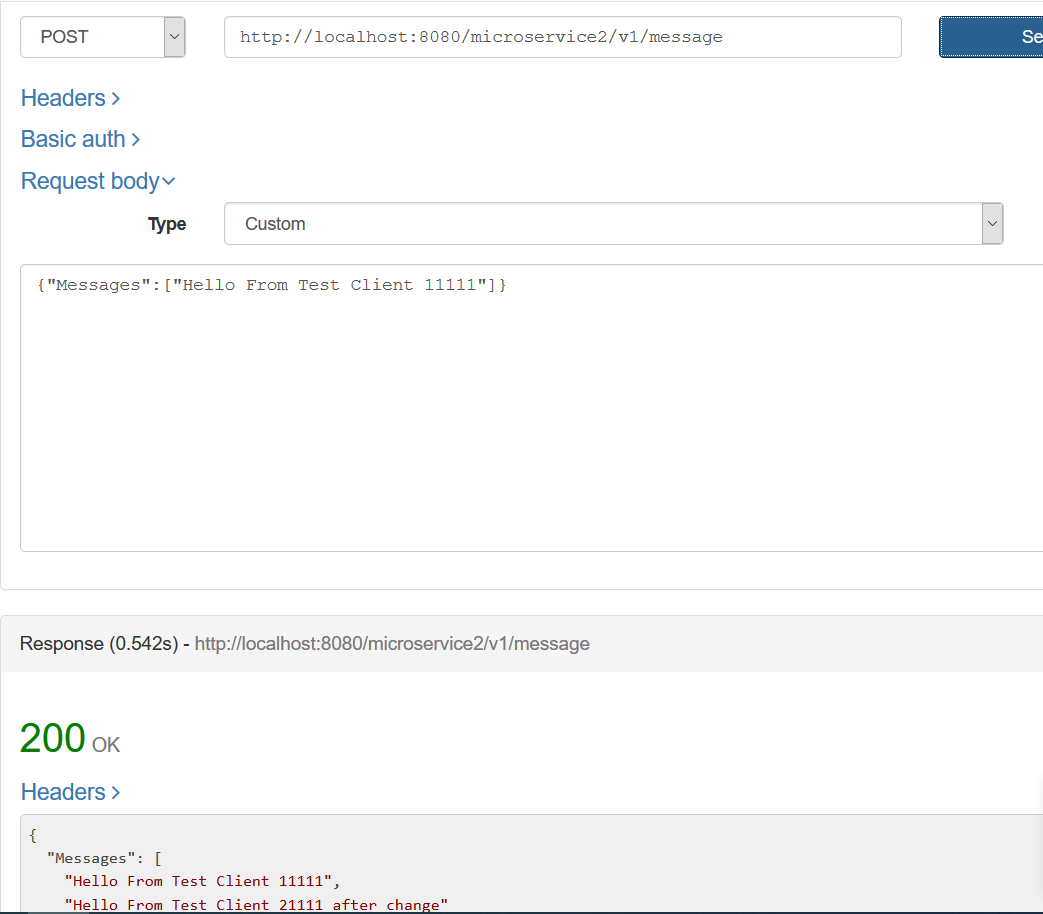
"Messages": [

"Hello From Test Client 1111",

"Hello From Test Client 2111 after change"

]

}



This second example shows that ACE Microservice 2 is designed to build up an array of messages.

POST to the following URL

http://localhost:8080/microservice2/v1/message

with data

{"Messages":["Hello From Test Client 1111"],["Hello From Test Client after change"]}}

You should get the following response returned

{

"Messages": [

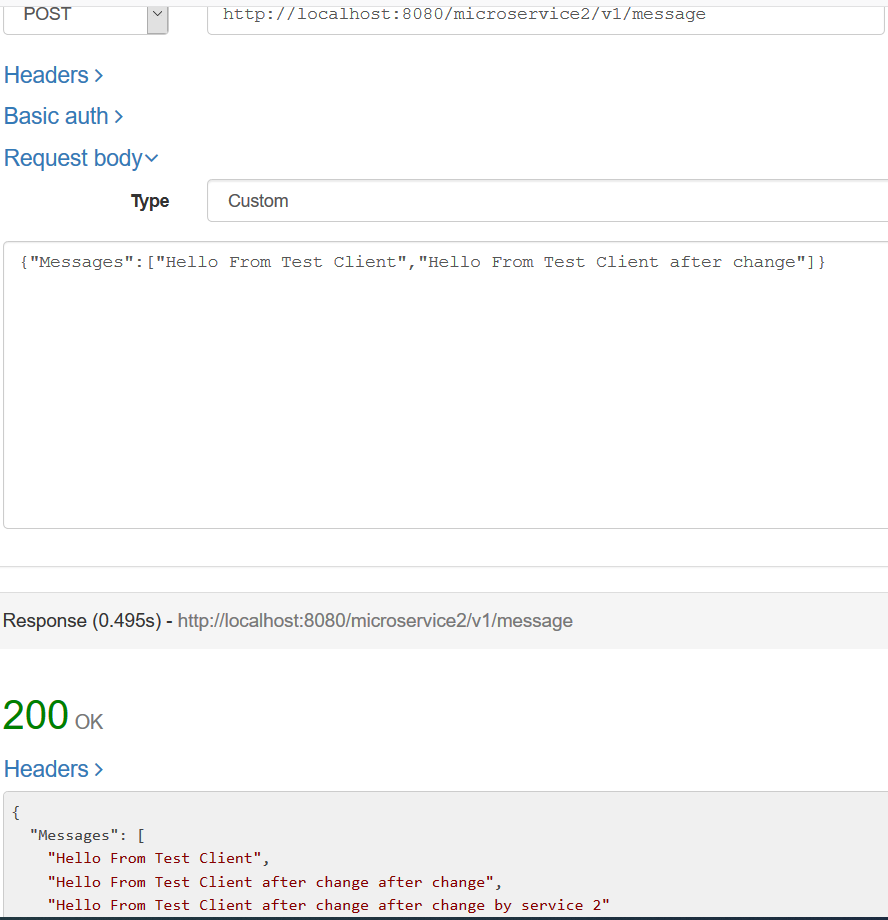
"Hello From Test Client 1111",

"Hello From Test Client after change"

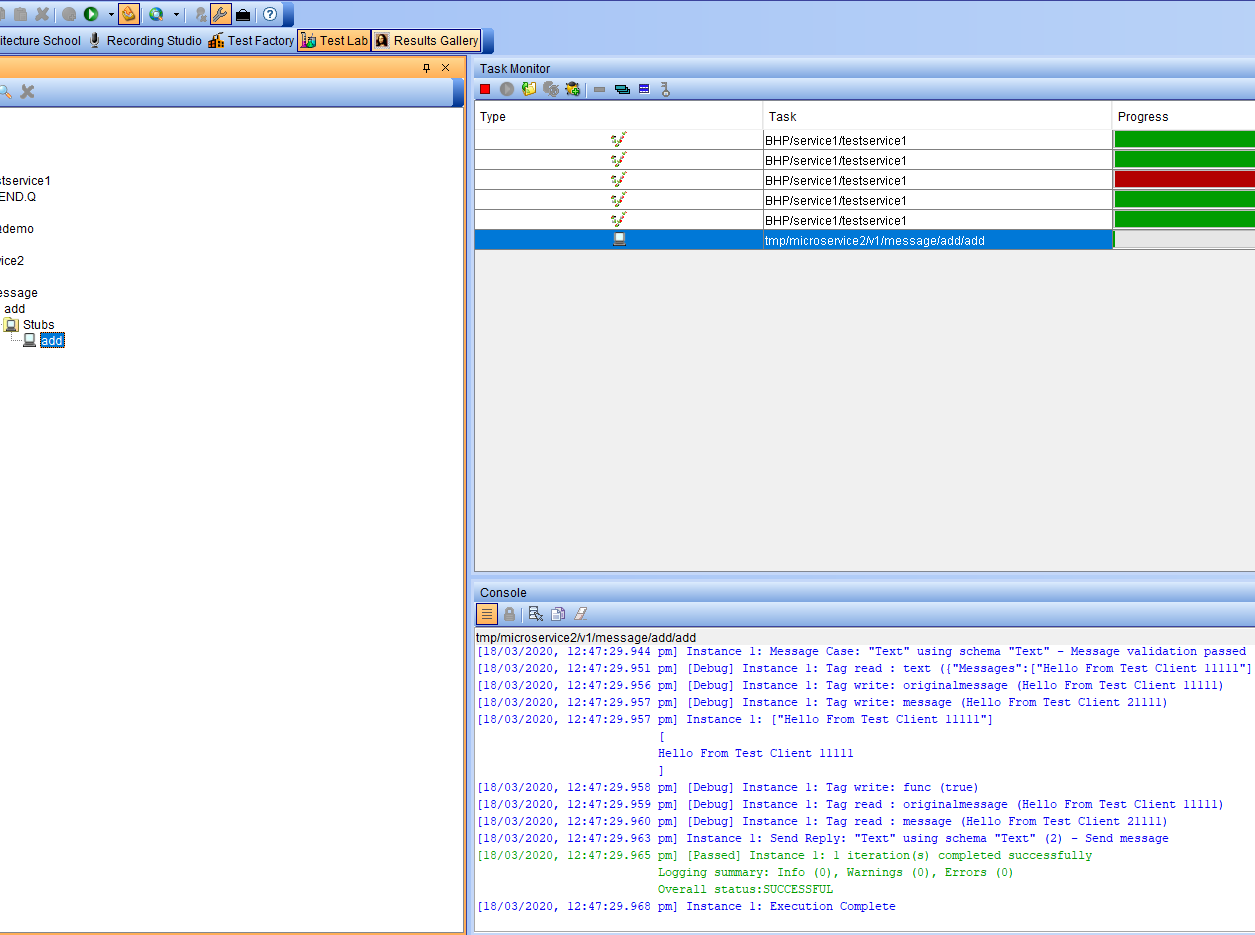
"Hello From Test Client after change after change"

]

}

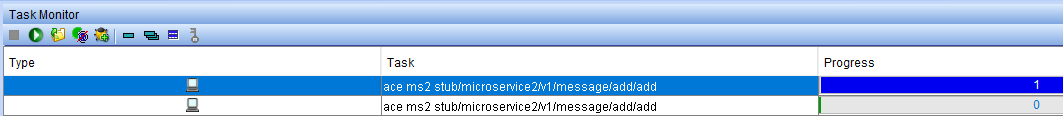


Switch back to the IBM Rational Integration Tester (IRIT) and review the Test Lab->console output for your tests

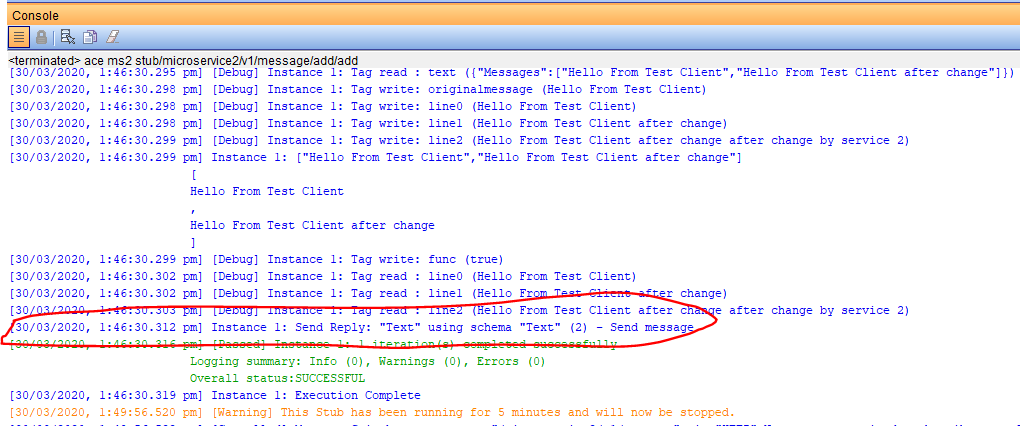


#### Check the results against pre-configured data

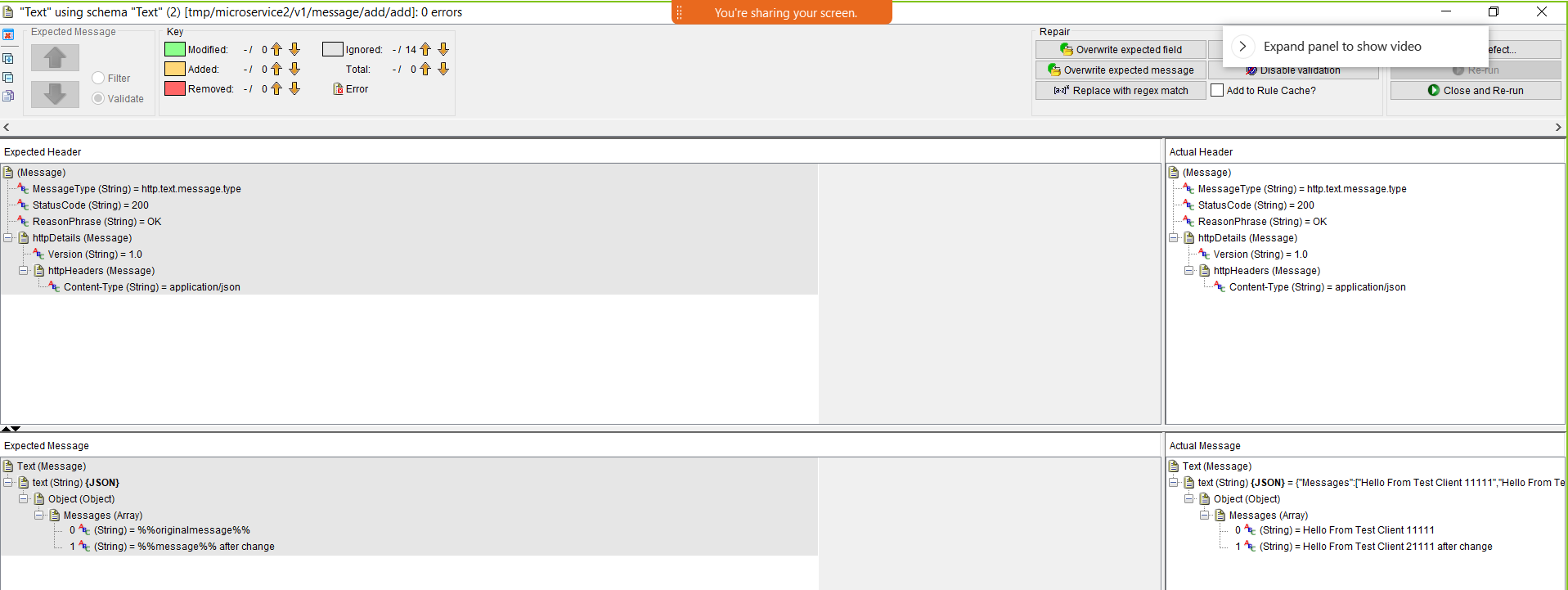
Select the Task for the test in the Task monitor



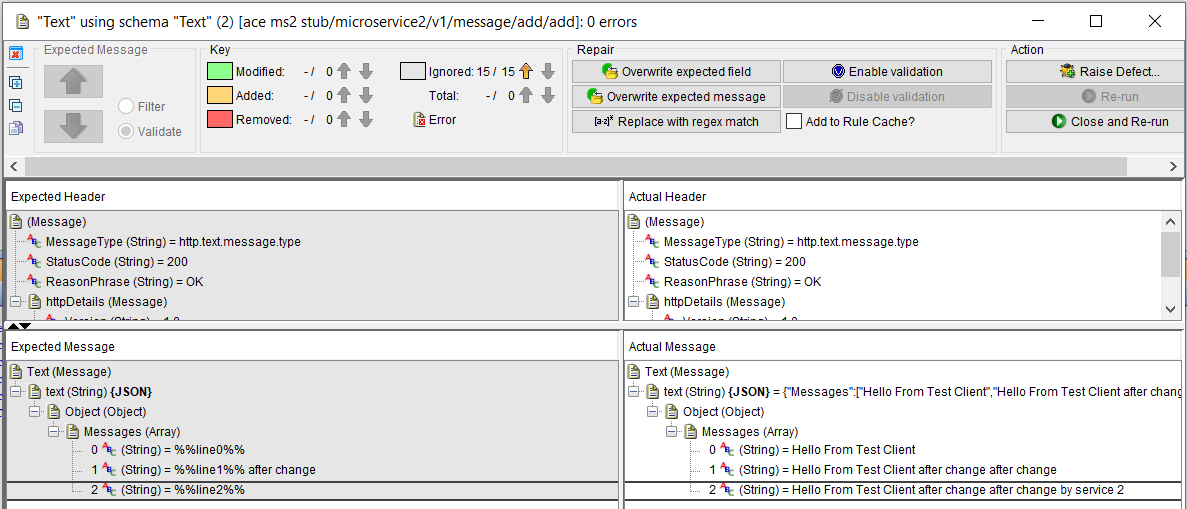
Click on the send reply line in the console – “Instance : Send Reply:’Text’ (2) – send message”



This brings up a comparison screen where you can compare the actual results with the expect results.



Example 2 results



### Testing the ACE Microservice 2 stub using IBM Integration Tester locally and via a

### Public IP address

The ACE microservice 2 stub will be running on your machine/laptop but will be called by ACE microservice 1 which is running on RH Openshift (in this example running on the IBM Public Cloud). Therefore, the ACE microservice 2 stub will need to be exposed with a public IP address.

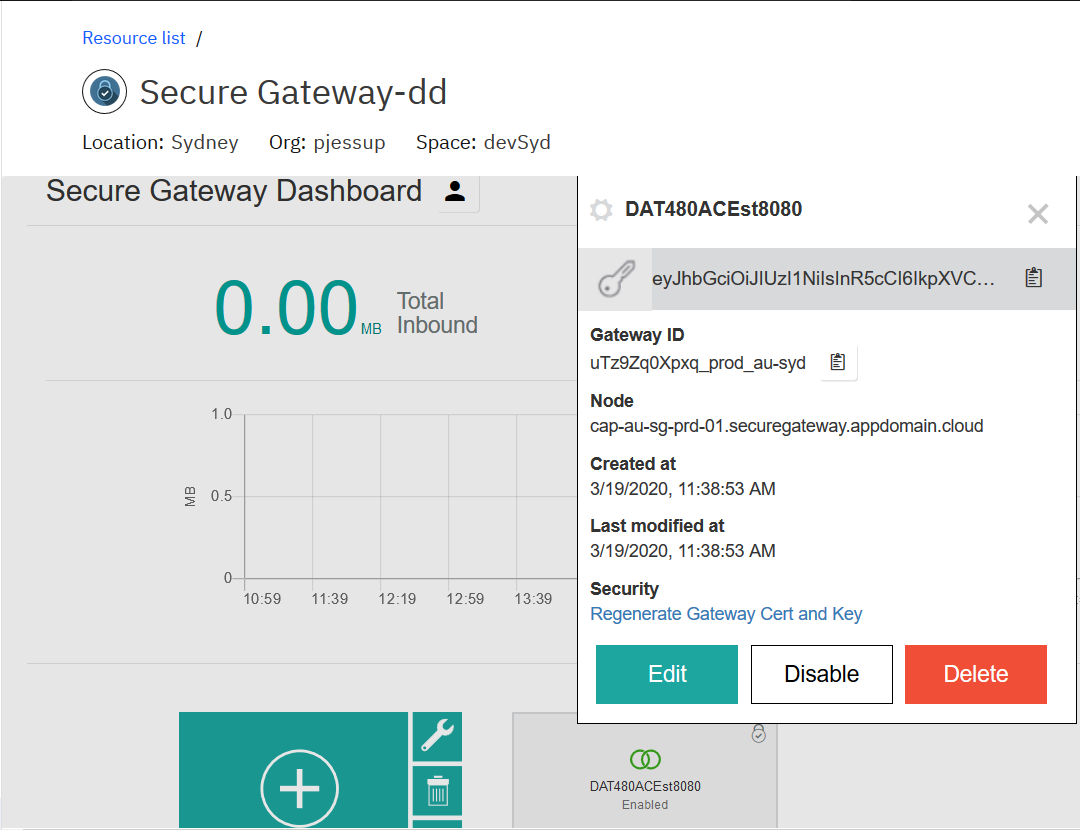
The IBM Secure Gateway Service can be used to enable a public IP to a local machine/laptop

### IBM Secure Gateway Service – Server side

Provision a free IBM Secure Gateway Service on the IBM Cloud

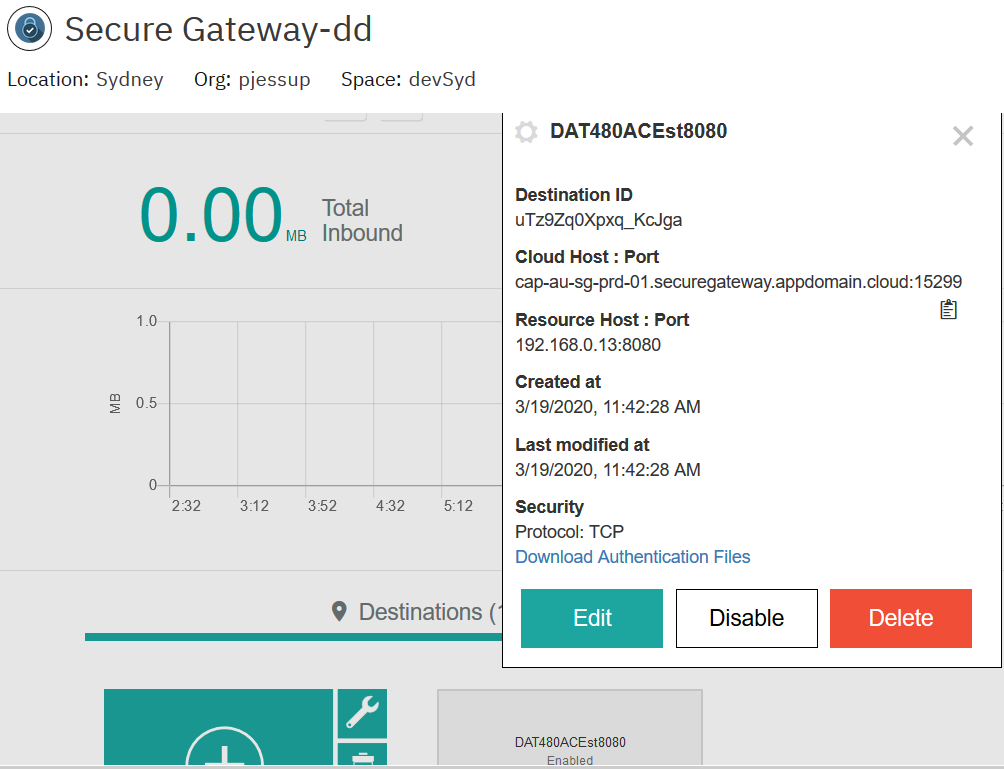
Follow the guide (part of the service) to:

* Create the server side gateway
* Download the client side component
* Configure a server side destination



During this process:

* Copy the gateway ID and token for use in the client side set up
* Open the destination and copy the public IP address and port number



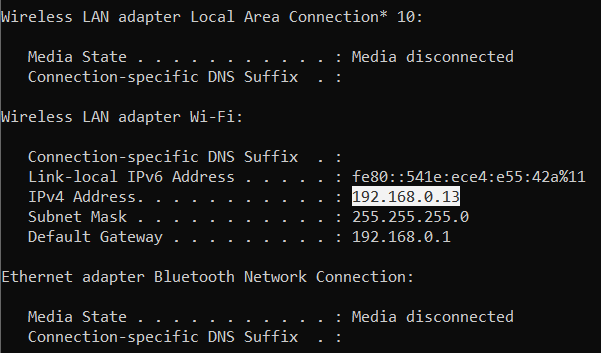
In our example:

cap-au-sg-prd-01.securegateway.appdomain.cloud:15299

### IBM Secure Gateway Client side

On the client side, once the client is downloaded and installed you will need the IP address that your machine is using at the time of set up (note this can change)

Use the ipconfig or similar command to find your IP address



It could be a wifi adapter, Bluetooth or IBM 9. Address ( it depends) + plus the port that IBM Rational Integration Tester has exposed for HTTP connections.

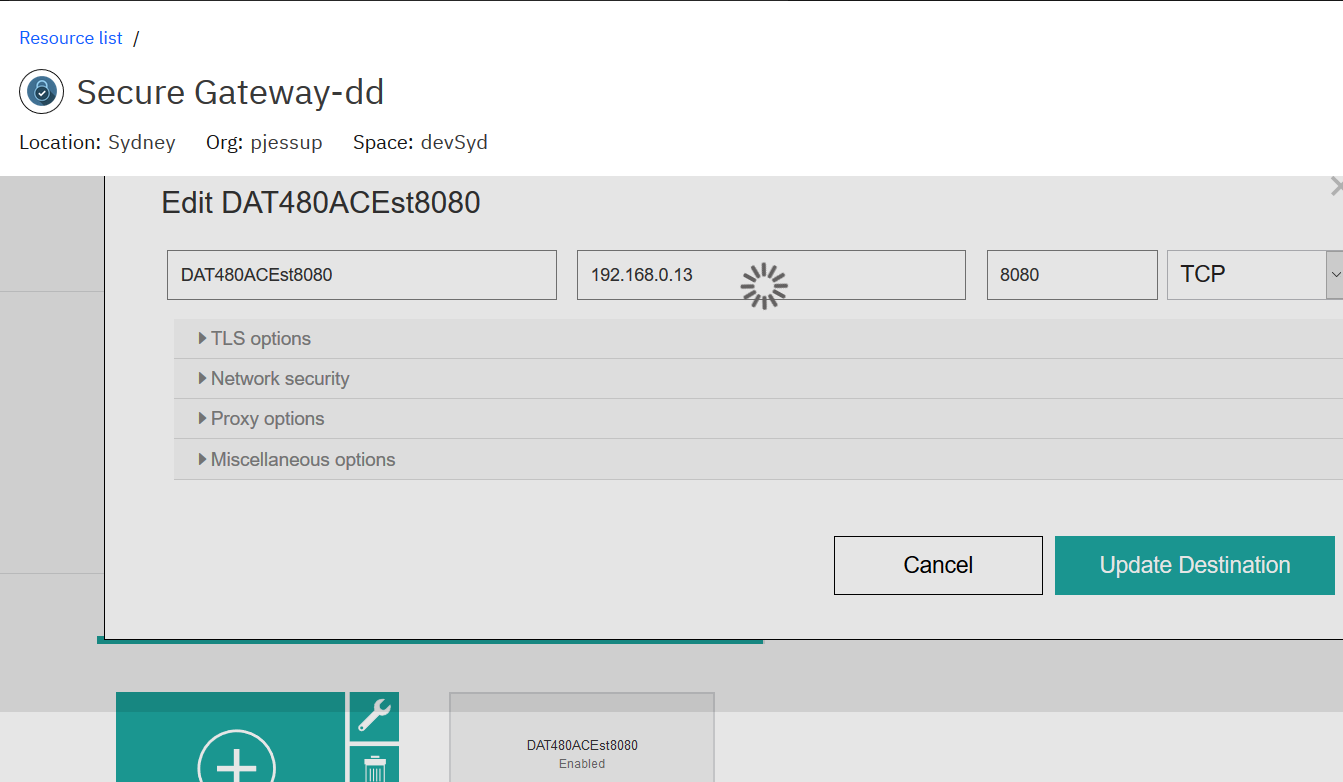
In my example the queue manager is listening on 1515 (used in a later tutorial) and my IBM Rationla Integration Tester (IRIT) stub for ACE MS2 is listening on 8080

The 192…. Address is my wifi adapter when I am connected home.

The 172.… Address is my Bluetooth when connected on via mobile phone.

The 9 …. Address is when I am at an IBM office

Returning to the Server side briefly, update the destination with the IPaddress and port number for your laptop/machine.



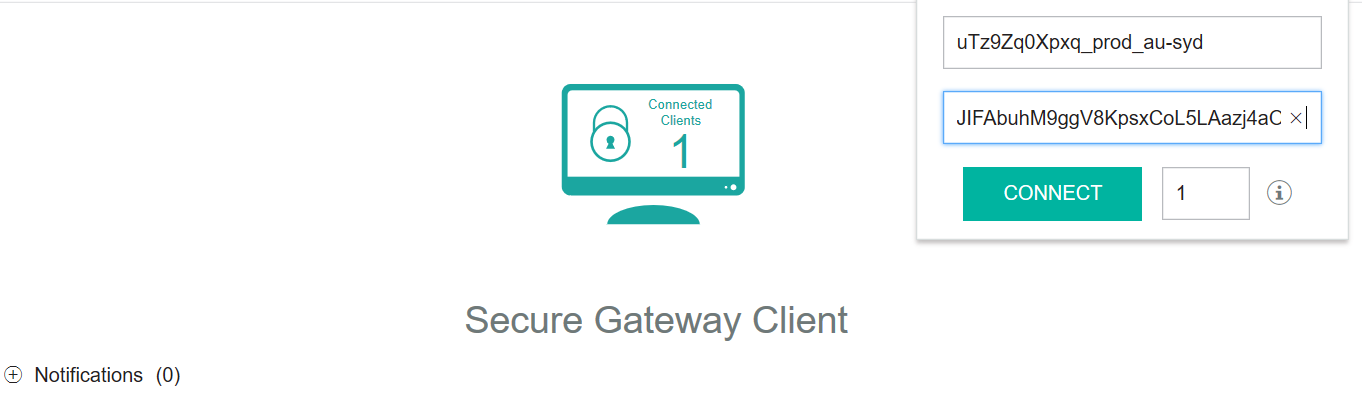
Back on the client side open the browser based Secure Gateway client configuration. Using the URL:

<http://localhost:9003/dashboard>

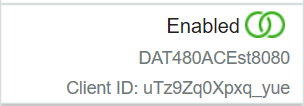
Add a connection.

Give it a name DAT480ACEst8080 for example

Use the copied gateway Id and token captured earlier from the server side



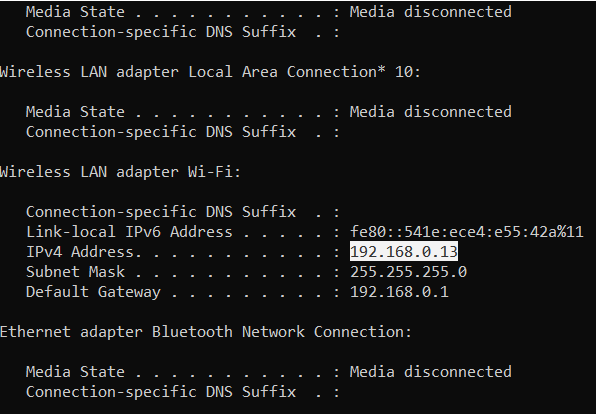
Click connect



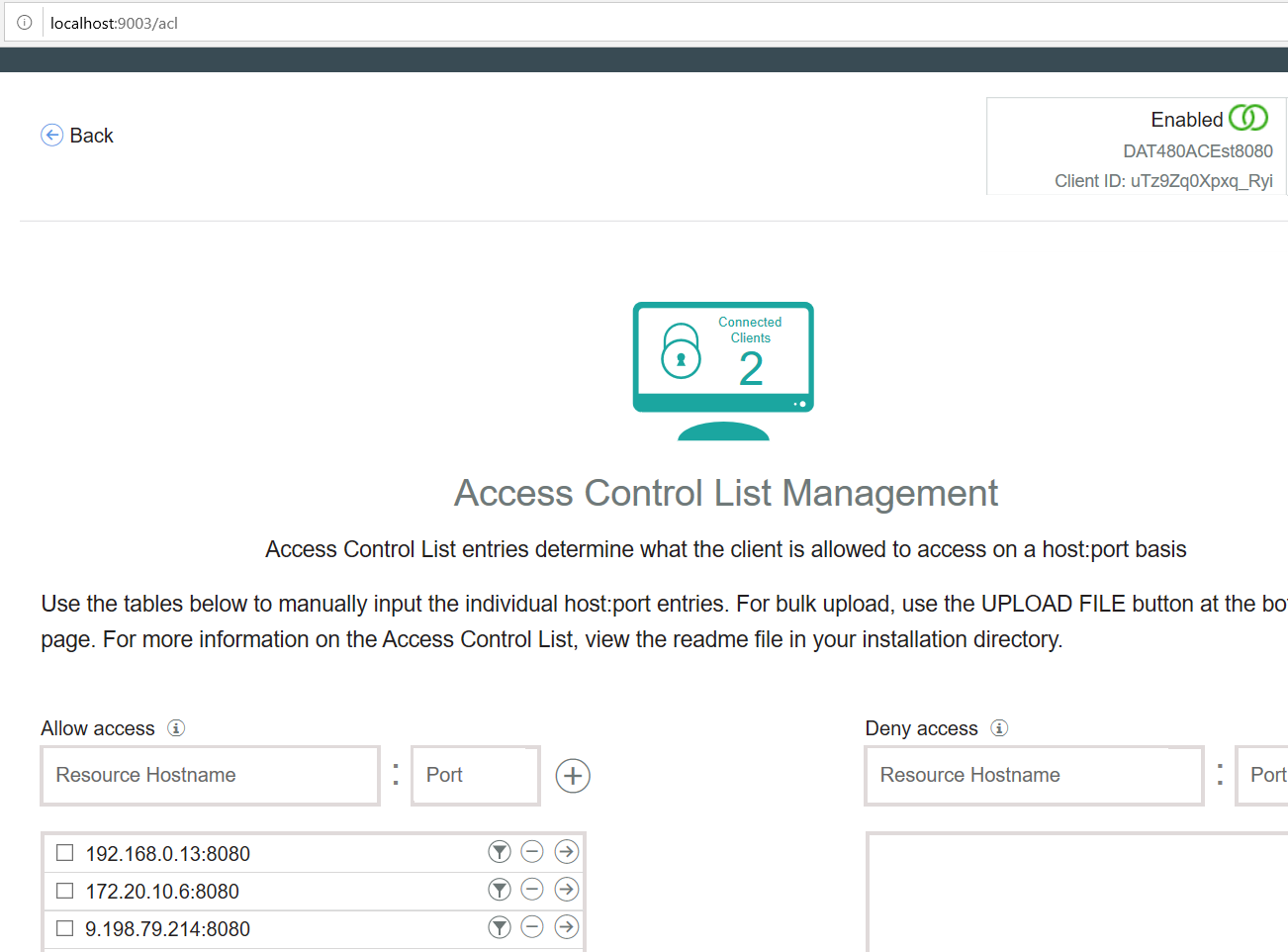
Next set up the ACLs for the inbound connection using the URL

<http://localhost:9003/acls>

Type IP config on your machine/laptop to find the IP address of the currently connected adapter

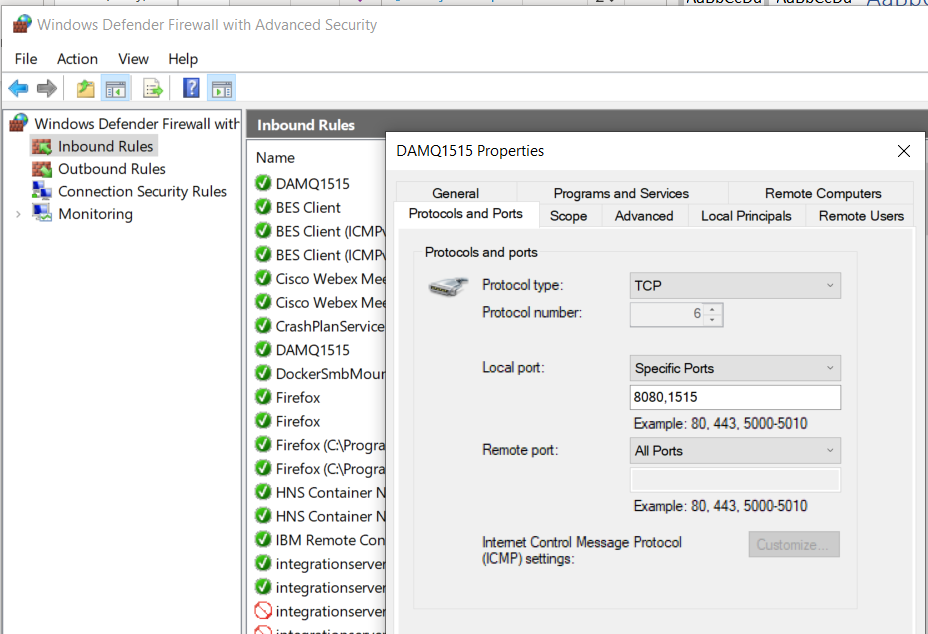


Configure the ACLs for “allow access” in the console for any IPAddress/port number combinations your laptop/machine might be known by.



### Windows firewalls

If running on Windows you may need to configure an Inbound Rule to allow traffic via certain ports.



### Test calling the RTVS ACE MS2 stub via public IP

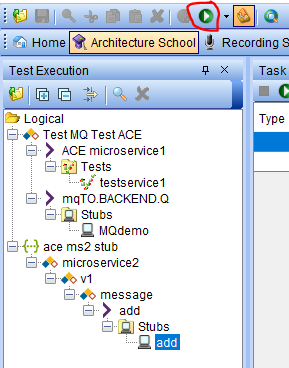
In IBM (Rational) Integration Tester we will need to restart the ACE MS2 stub

Select the Test Factory OR Test Lab tabs

Expand the Logical file hierarchy

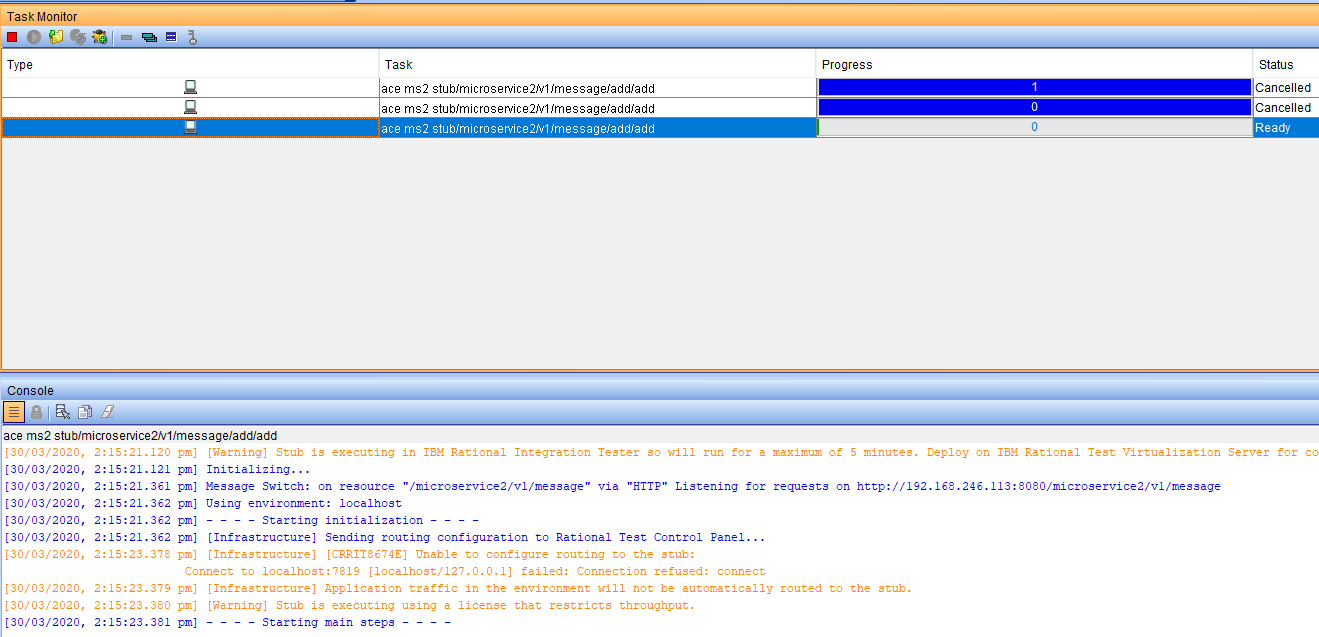
Ace ms2 stub->microservice2->v1->message->add->stubs->add

Double click on “add” then hit the circular green “Play” button to start the ACE MS 2 stub running



In the Task Monitor pane select the Task run for the stub. It will have a status of ready.

You also see the stub output in the console pane.



The IBM Secure Gateway will be offering a public IP address to connect to your laptop/machine. In my example it is: cap-au-sg-prd-01.securegateway.appdomain.cloud:15299

Build out the URL to call an ACE MS2 by adding the base URL to the IPAddress port number:

cap-au-sg-prd-01.securegateway.appdomain.cloud:15299/microservice2/v1/message

Use a REST Client to

POST

http://cap-au-sg-prd-01.securegateway.appdomain.cloud:15299/microservice2/v1/message

with data

{"Messages":["Hello From Test Client 1111"],["Hello From Test Client 2222"]}

The REST client should receive the response:

{

"Messages": [

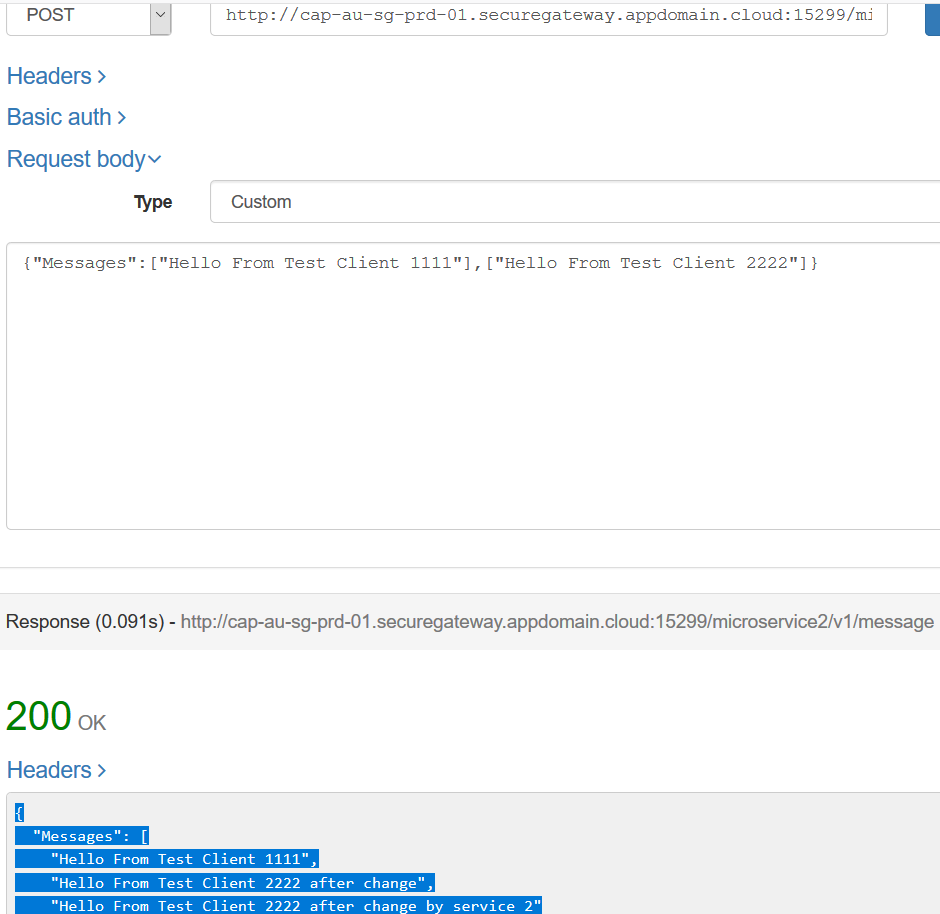
"Hello From Test Client 1111",

"Hello From Test Client 2222 after change",

"Hello From Test Client 2222 after change by service 2"

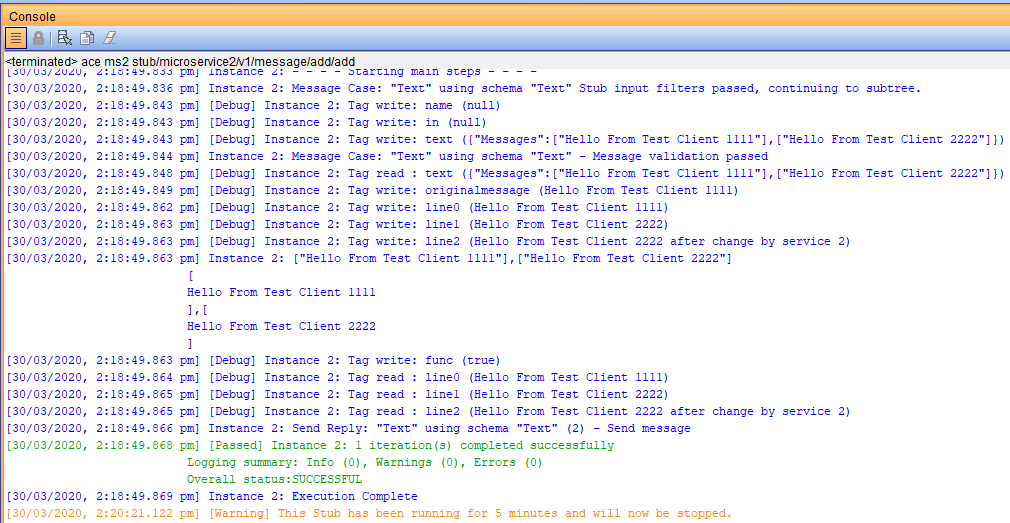
]

}



Return to IBM (Rational) Integration Tester and review the console.

You should see a success execution of the stub.



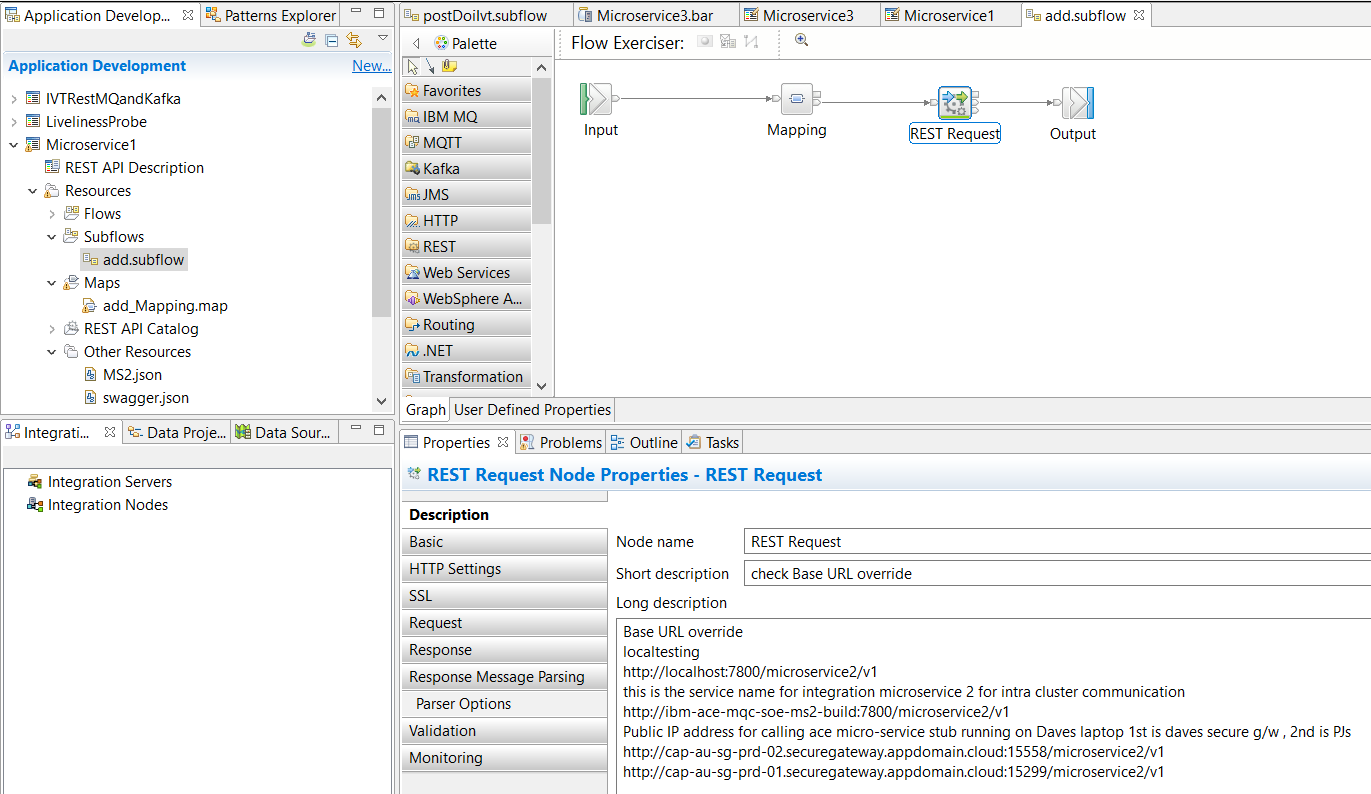
## Use IBM Integration Tester to test ACE Microservice 1 (on RH Openshift) connected to ACE Microservice 2 Stub

For this test ACE Micro Service 1 running on RH Openshift will have its RestRequest node configured with the URL it needs call the ACE Micro Service 2 stub running on IBM (Rational) Integration Tester via the public IP address offered by the IBM Secure Gateway service.

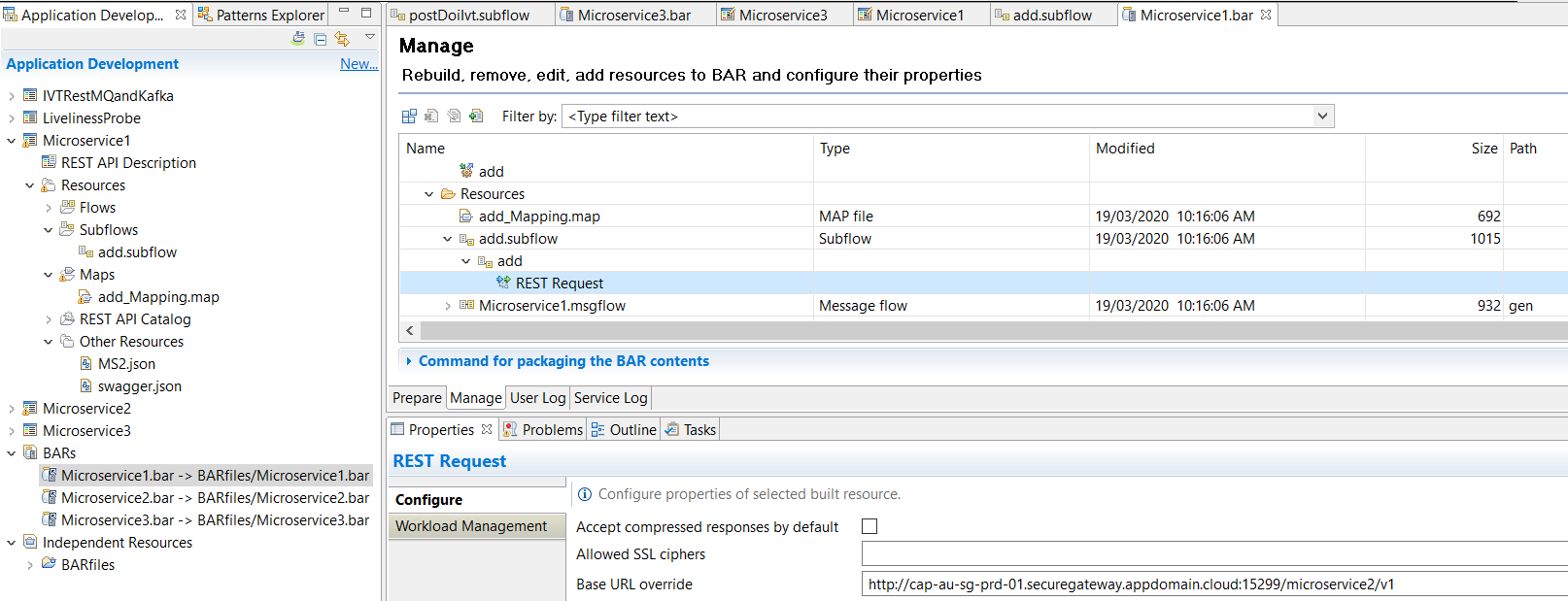
In this example that URL is:

http://cap-au-sg-prd-01.securegateway.appdomain.cloud:15299/microservice2/v1

In the IBM App Connect Enterprise Toolkit



Rather than changing the “source code” in the add.subflow subflow. Use the BAR override parameters on the BAR file itself to update the URL that ACE MS1 will use to call ACE MS2 to direct it to the stub on IBM (Rational) Integration Tester.



Next, the image on RH OpenShift will need to be replace with a new version containing the updated BAR. The document “**1.Developer Experience for ACE MQ with RHOS Tools and Tekton v1.1**”

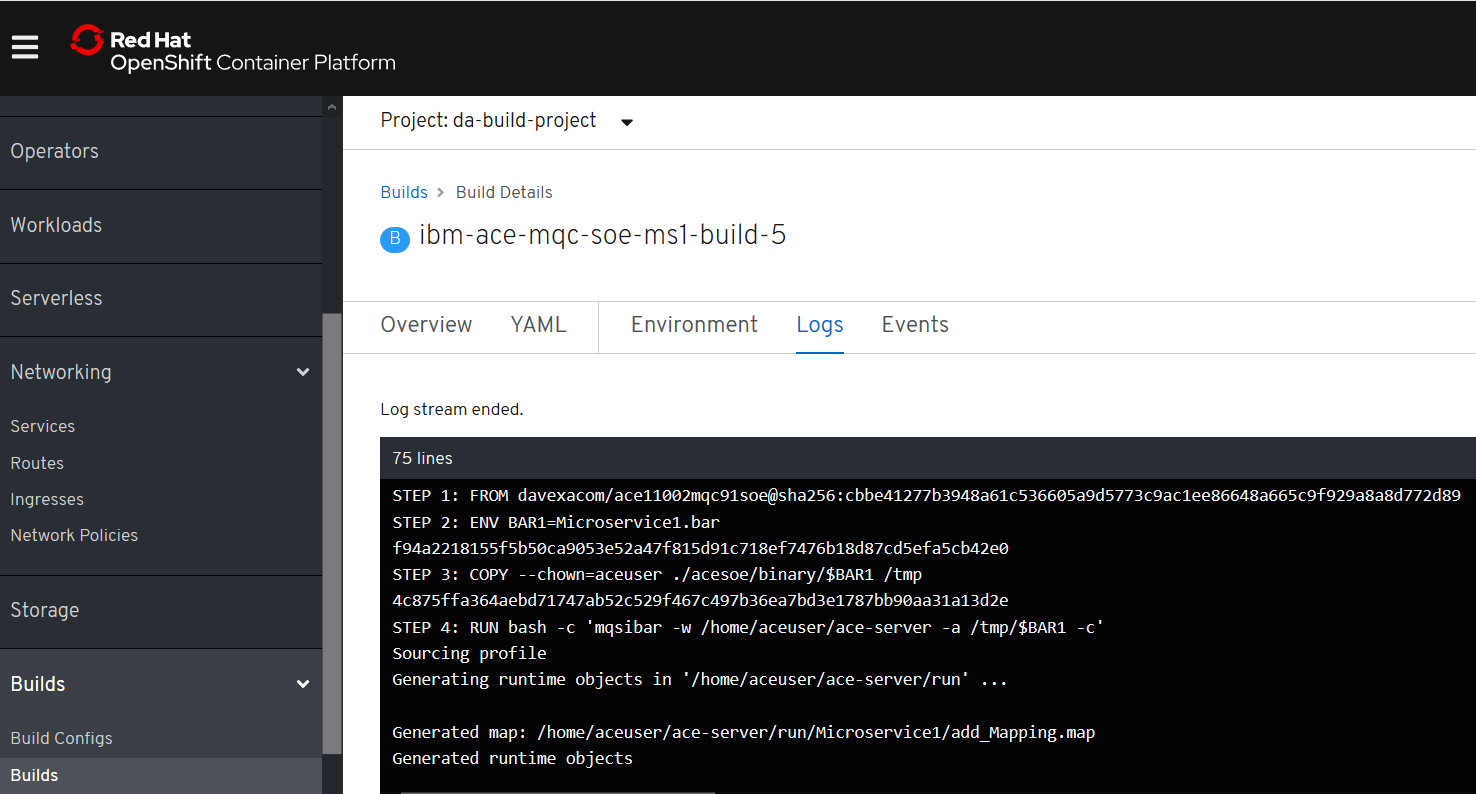
Section: **Build and Run ACE Microservice 1 from SoE on RH OpenShift**

Page : **49**

Describes the following steps to rebuild,deploy and run the updated ACE MS1, if you have not already set up ACE MS1.

The steps are

1. Save the BAR file
2. Push it to the GIT repos
3. Webhook fires and RHOS will build and deploy a new container with the updated BAR file



Check the logs on the triggered build/rebuild to ensure the build was successful

RH OpenShift console->Builds->builds->ibm-ace-mqcsoe-ms1-build-n

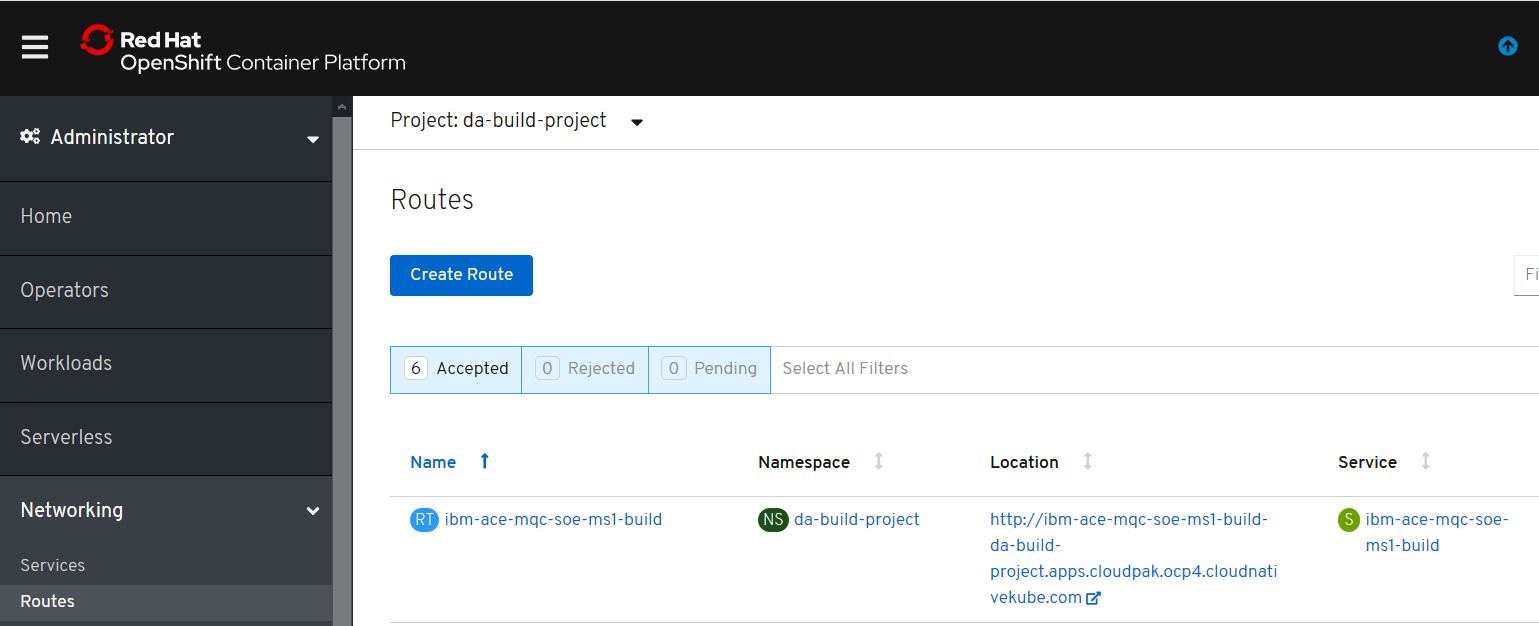
Check the logs on the ACE MS1 pod to ensure the latest version is running

RH OpenShift console->Workloads->pods->ibm-ace-mqcsoe-ms1-build-n-xnxnxn->logs

ACE MS1 on RH OpenShift is now pointing (via the secure gateway) to the stub of ACE MS2 on IBM Rational Integration Tester.

### Get the ACE MS1 route details from RH OpenShift console

RH OpenShift Console->Networking->routes->ibm-ace-mqc-soe-m1-build

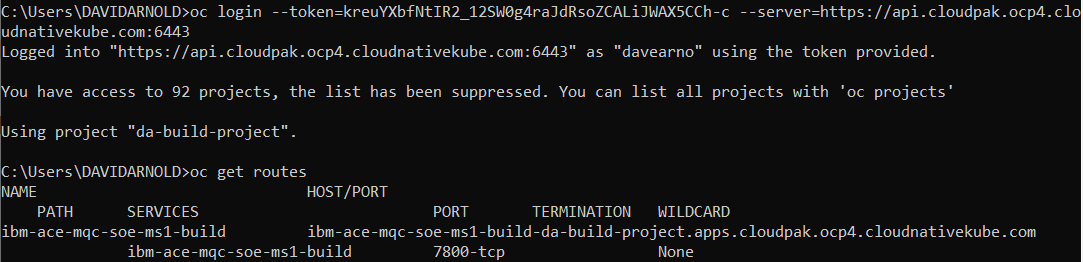


Or use

Oc get routes

To retrieve the target hostname:

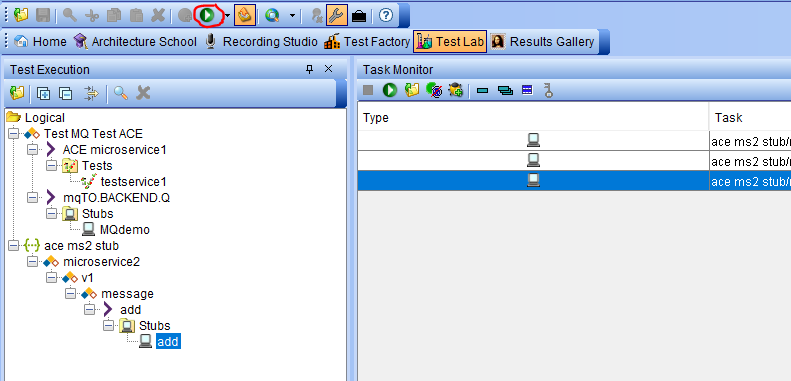
ibm-ace-mqc-soe-ms1-build-da-build-project.apps.cloudpak.ocp4.cloudnativekube.com



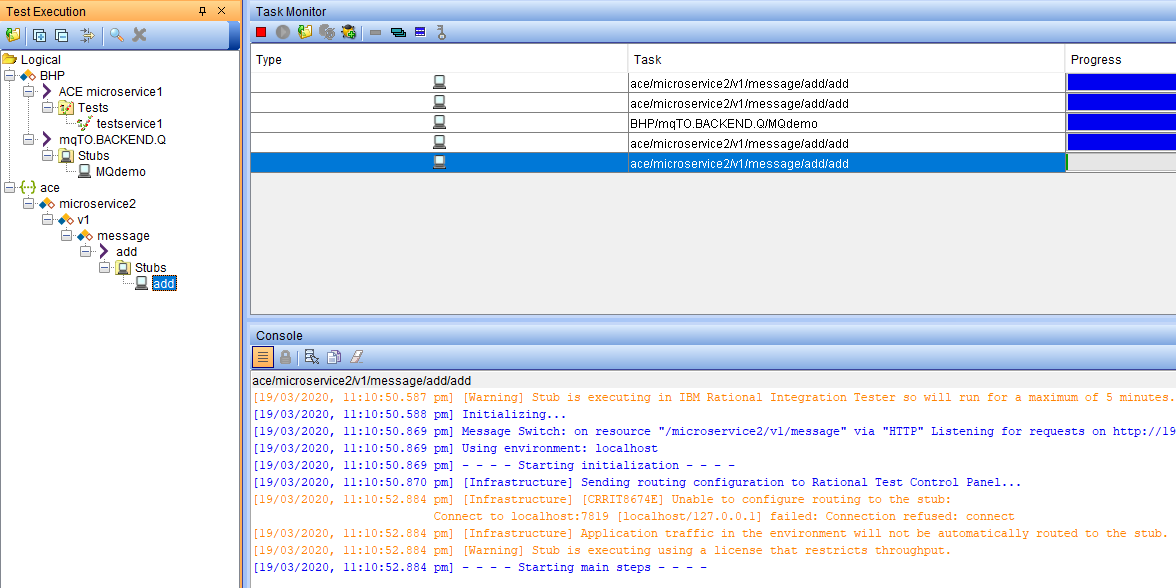
Add the ACE MS 1 base URL to the hostname and then using a REST Client to ensure you have connectivity to ACE MS 1 and the ACE MS2 stub it is set up to call.

### Start/Restart the IBM Rational Integration Tester ACE MS2 Stub

Note: stubs in IBM Integration Tester only run for 5 mins before needing a restart via the green “play” button



Check the console output to observe the stub is up and running



### Use a REST client to call ACE MS1 (which in turn called ACE MS2 stub)

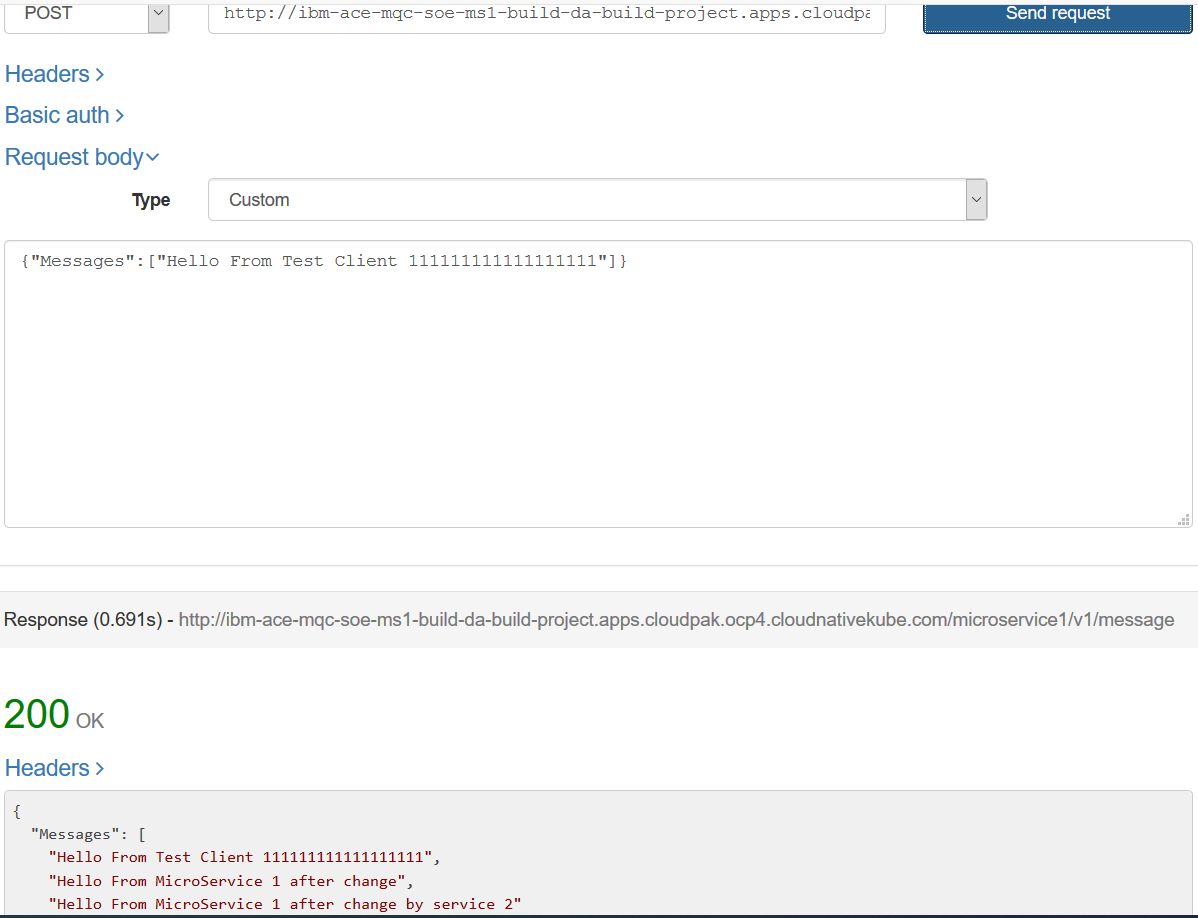
Use a local REST Client to test.

POST to URL:

<http://ibm-ace-mqc-soe-ms1-build-da-build-project.apps.cloudpak.ocp4.cloudnativekube.com/microservice1/v1/message>

with Data

{"Messages":["Hello From Test Client 111111111111111111"]}

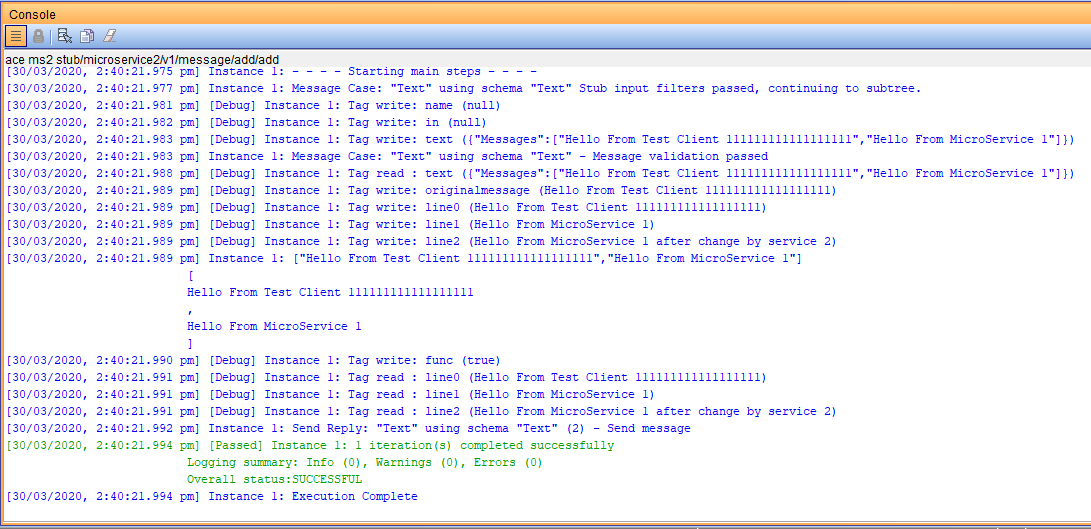


The response should be:

{"Messages":["Hello From Test Client 111111111111111111"],["Hello From microservice after change"],["Hello From Microservice 1 after change by service 2"]}

### Observe results in ACE MS 2 stub console

The IBM Rational Integration Tester console should reflect a successful test



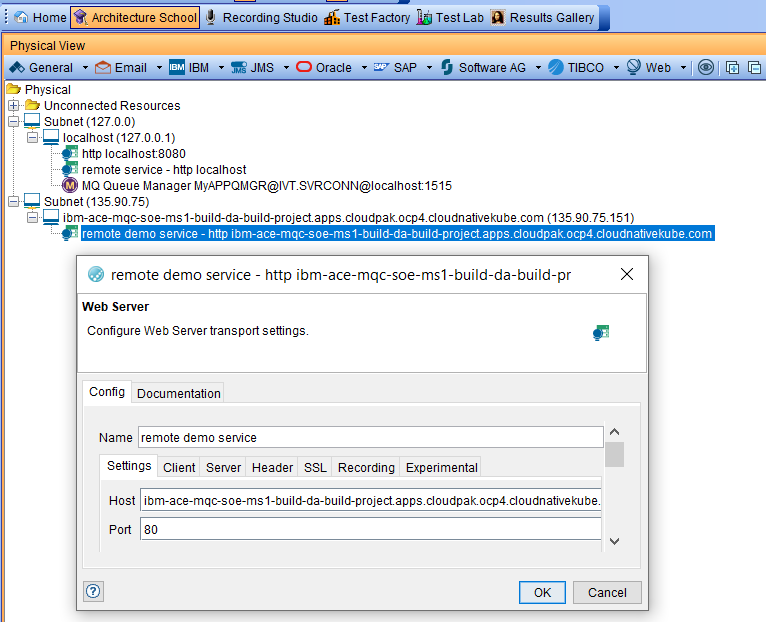
### Invoking ACE MS1 on RH OpenShift from a IBM Rational Integration Tester

Next we will use IBM Rational Integration Tester to test ACE MS1 on RH Openshift, where it will call the ACE MS2 stub running back on IBM Rational Integration Tester.

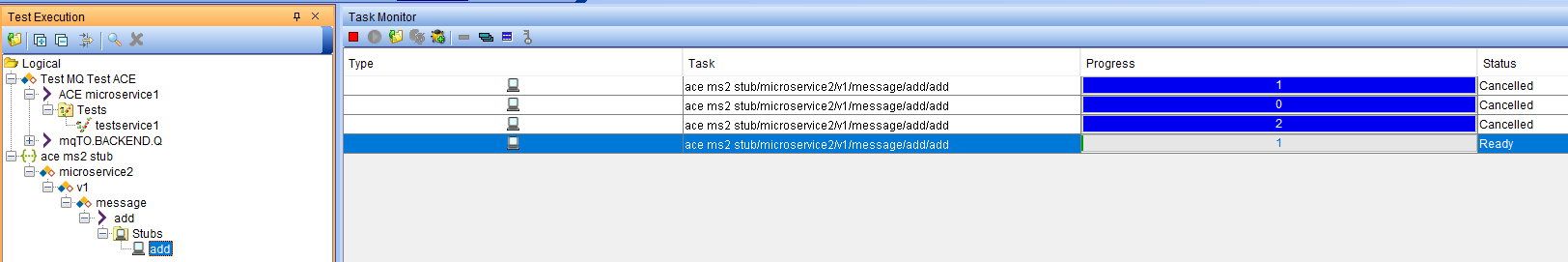
Review the connectivity details for ACE MS1 in IBM Rational Integration Tester->Architecture School->Physical View.

Navigate to subnet 135.90.75 (i.e. not the local host subnet).

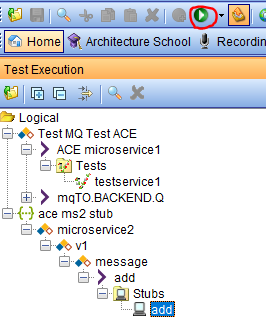
In the settings for the remote demo service ensure you have the ipaddress for ACE MS1 on RH Openshift. For example ibm-ace-mqc-soe-ms1-build-da-build-project.apps.cloudpak.ocp4.cloudnative.com



Back in Test Lab in IBM Integration Tester, ensure the the ACE MS 2 stub is still running



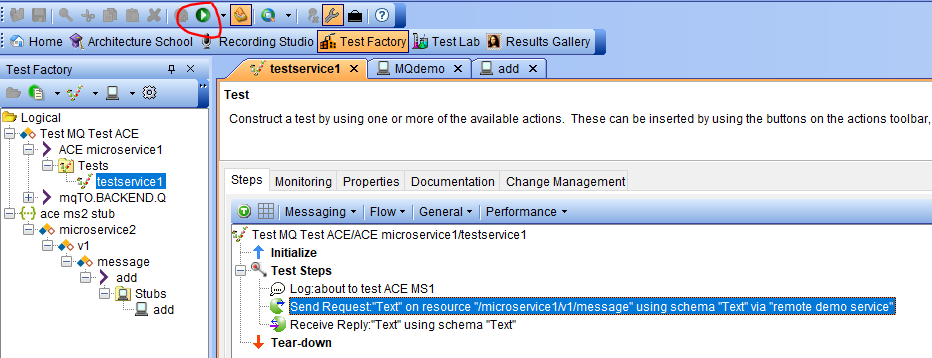
If not, select ace ms2 stub->ace microservice2->v1->message->add->stubs->add and start a new test run using the green “Play” button



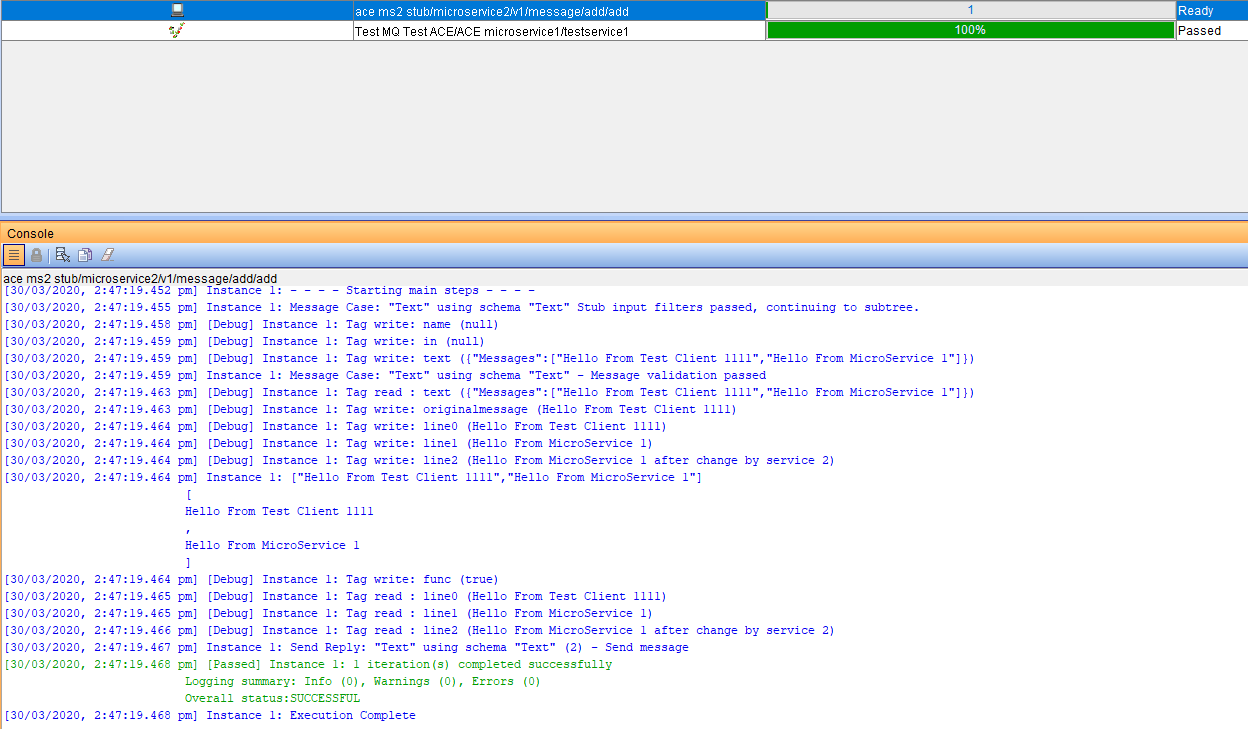
Next we will run the ACE microservice 1 test by navigating to Test MQ Test ACE->ACE microservice1->Tests

Select testservice1 then

Hit the green “play” button to run the test

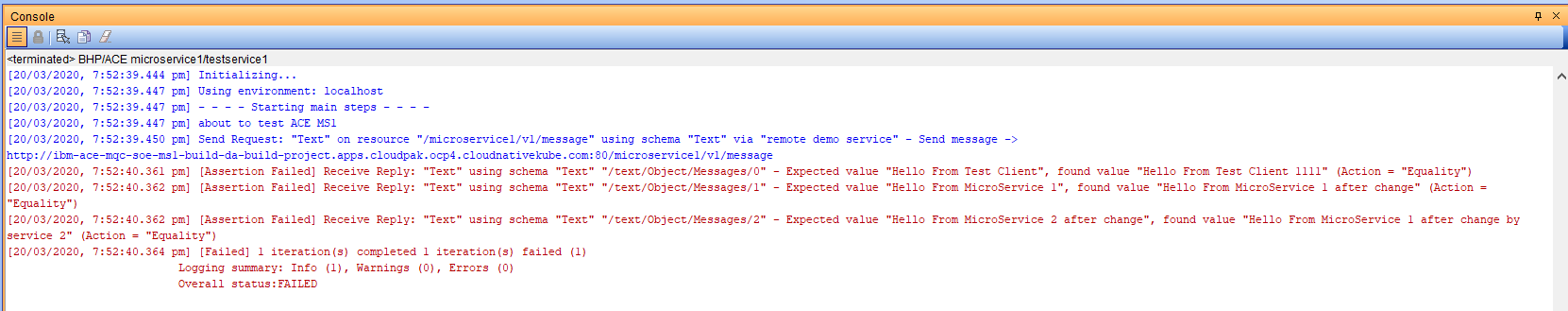


### Observe the results in the console for the ACE MS2 stub

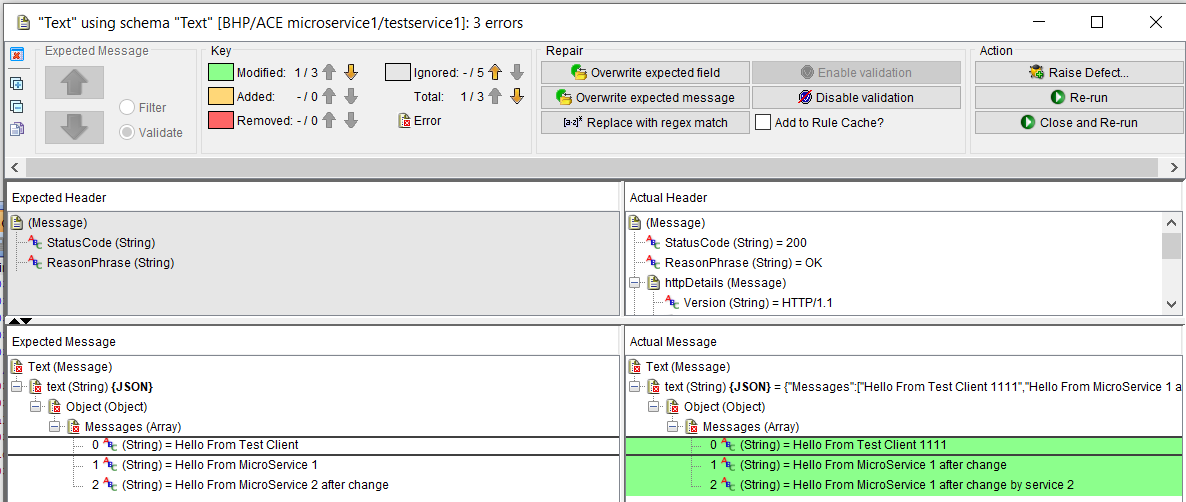


### Observe results in the console for the testservice 1

#### An example of a working test with incorrect data (unmatching data)

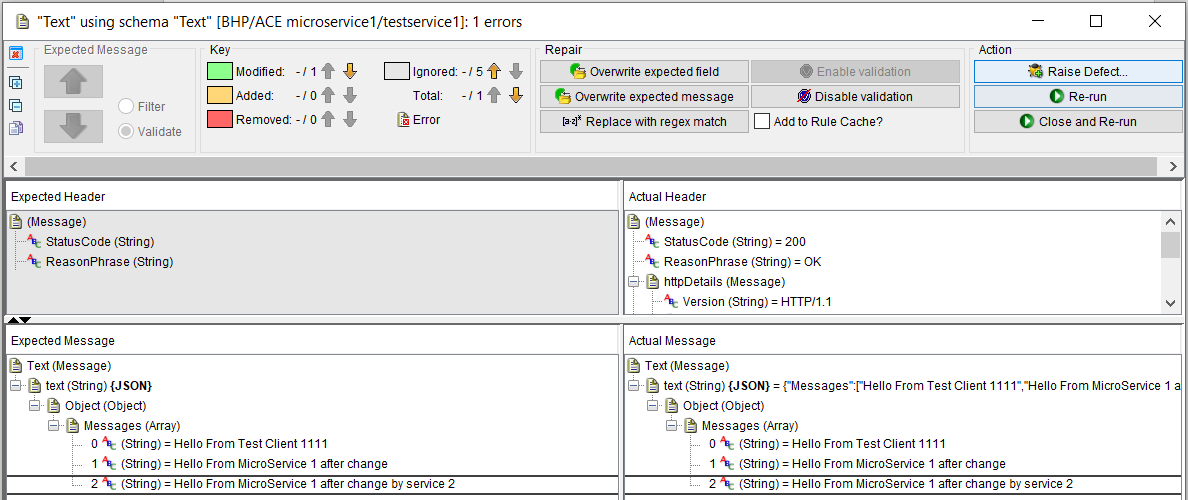


The test actually worked but we can compare the actual and expected

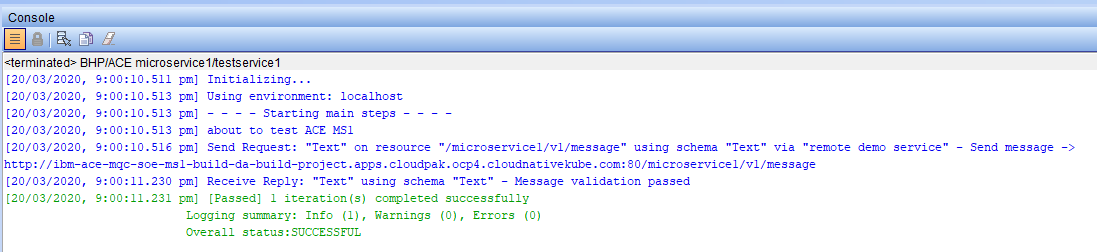


Note the semantics are correct but the data is different

Right click on the right and select overwrite expected

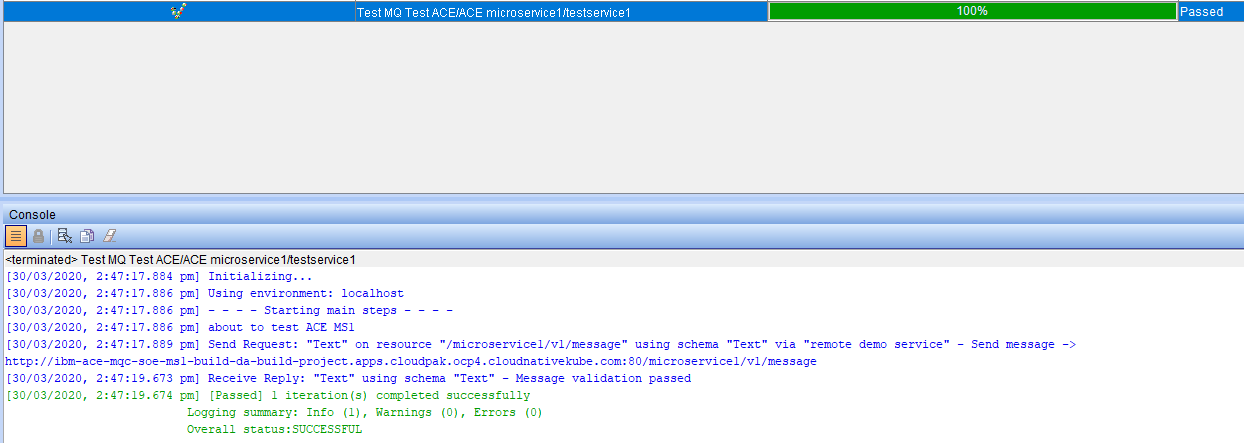


Re-run the test





#### An example of a working test with correct data (matching data)



# 2.Run and test ACE microservices deployed on RH OpenShift that connect to MQ

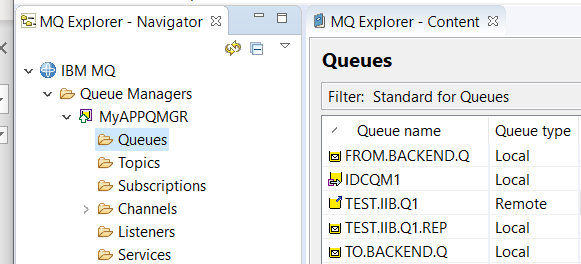
## Using ACE Micro Service 3 (MS3) with a remote MQ Queue Manager

### Configure and test an MQ Queue Manage on your local IBM Integration Tester

Create an MQ Queue Manager called MyAPPQMGR

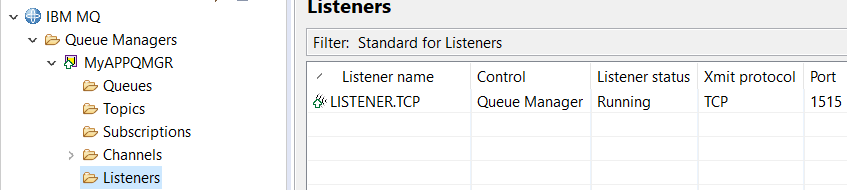
With queues FROM.BACKEND.Q and TO.BACKEND.Q

#### Queue Manager MyAPPQMGR and Queues

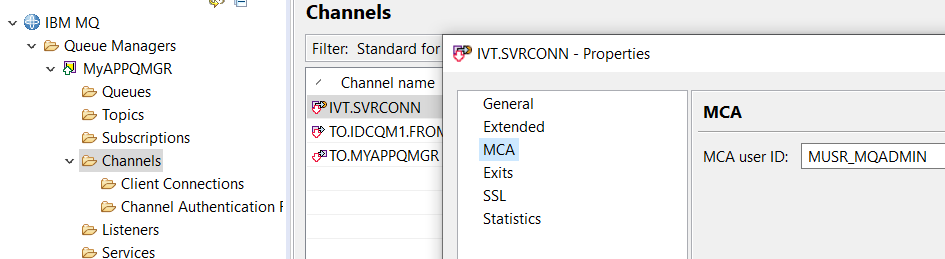


#### MQ Listener

An MQ Listener that listens on port 1515



### MQ SVRCONN Channel creation for Linux based queue manager



DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN) REPLACE

SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)

SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(\*) MCAUSER('mqm')

REFRESH SECURITY TYPE(CONNAUTH)

### MQ SVRCONN Channel creation for windows based queue manager

DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN) REPLACE

SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)

SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(\*) MCAUSER('MUSR\_MQADMIN')

REFRESH SECURITY TYPE(CONNAUTH)

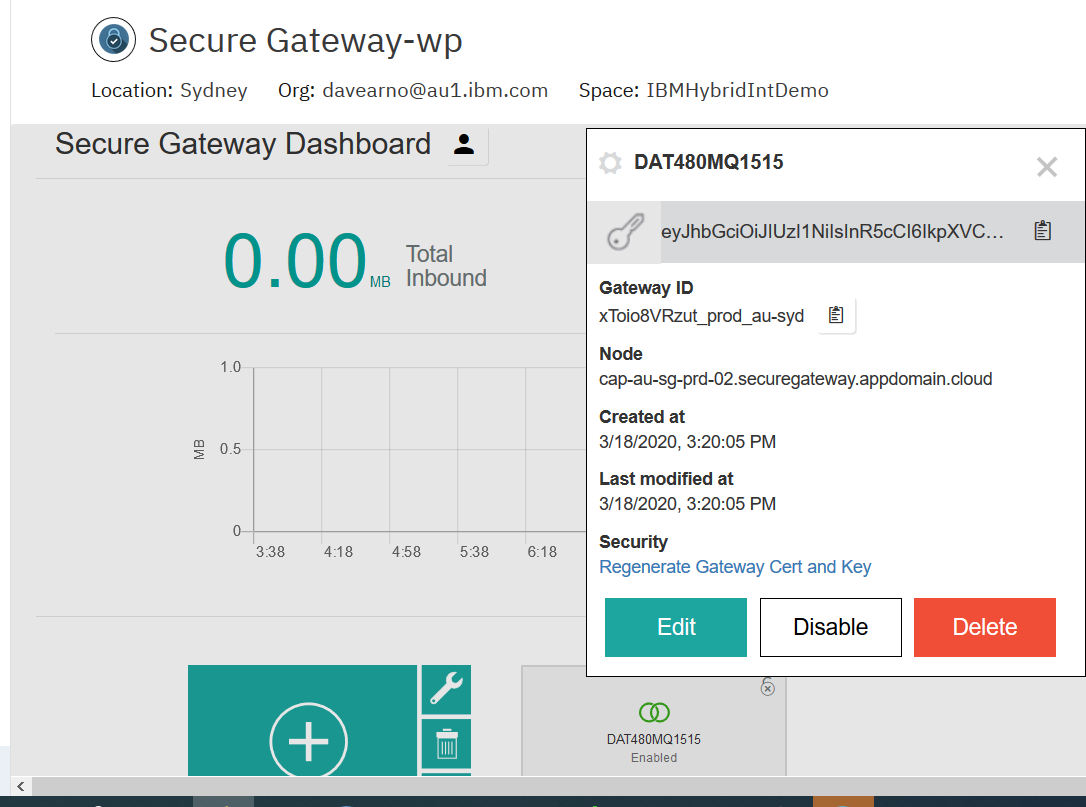
## Set up IBM Secure Gateway Service for Public IP to Queue Manager MyAPPQMGR

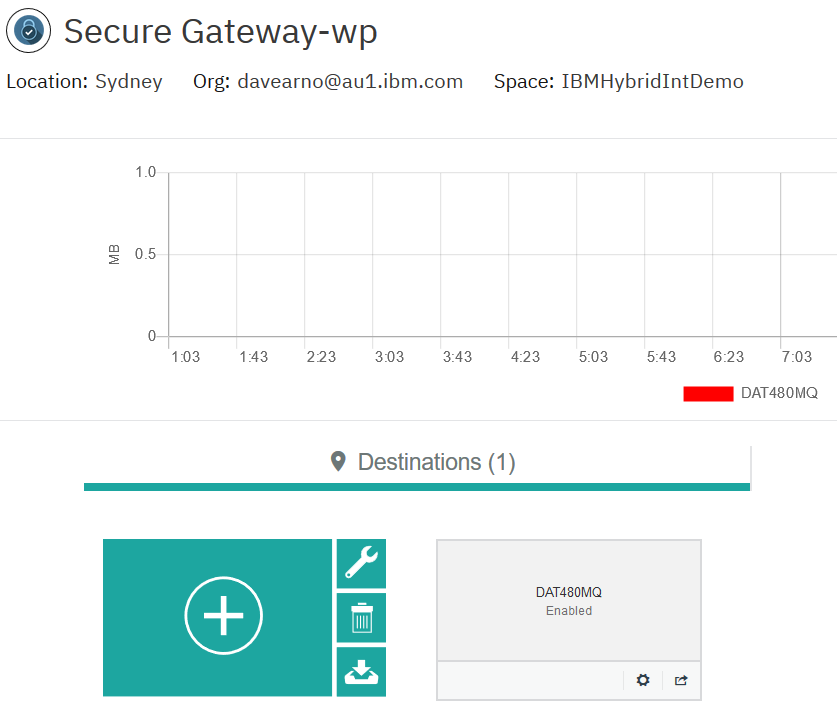
As with the ACE Microservice 1 and ACE Microservice 2 tests and stubs in the earlier section, ACE microservice 3 will require a public IP address to access an MQ Queue Manager that is set up on you machine/laptop.

The process is identical to that described earlier. You will need to create a new gateway and destination on the IBM cloud server side (or reconfigure the existing gateway to talk to MQ on port 1515 instead of to the IBM Integration Tester HTTP listener on port 8080)

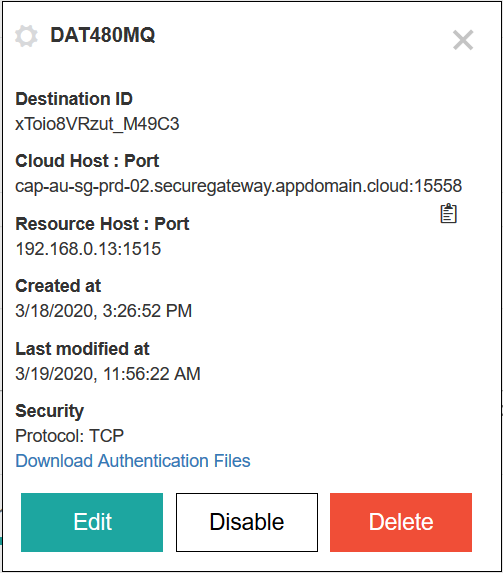
### IBM Cloud Secure Gateway - Server side

Copy the Gateway ID and the Token for use in the IBM Secure Gateway client side set up





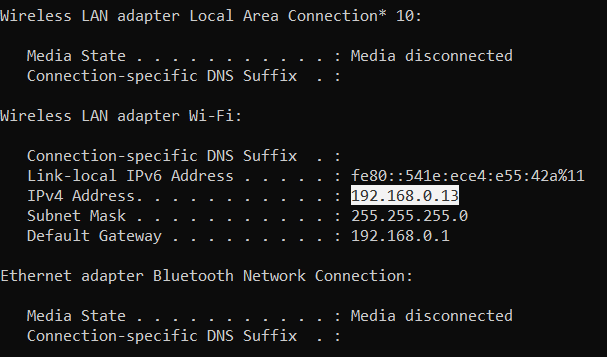
Copy the Cloud:Host Port number combination from the IBM Secure Gateway Service destination



This will be used on to override the ACE MS3 BAR parameters for the MQOutput node connection details in the microservice’s message flow.

### IBM Secure Gateway Service – Client (Laptop end)

Use ipconfig command to find the IP address of a connected network adapter on your machine/laptop.



Add it to the ACLs in the secure gateway client side console via URL:

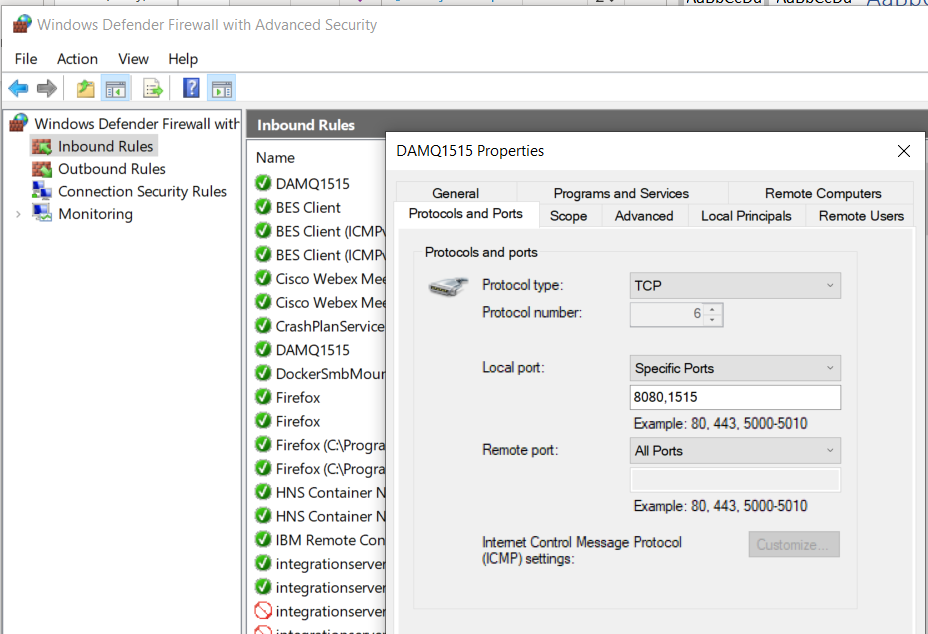
<http://localhost:9003/acl>

In my example, 192.168.0.13:1515 will be added to the “Allow Access” section



### Windows firewall

Again, as in the previous scenario, if running on windows you may need to create or update an inbound rule to add the port 1515 to allow connection to the MQ Listener.



## Using/Matching Secure Gateway parameters on ACE MS 3 MQOutput configuration

Depending on whether you have completed Part 1 of these materials you will need to check and potentially change the configuration of ACE MS3 running on RH OpenShift and trigger a build/rebuild.

See GitHub repo: https://github.com/DAVEXACOM/tekton-ace-example/tree/master/doc

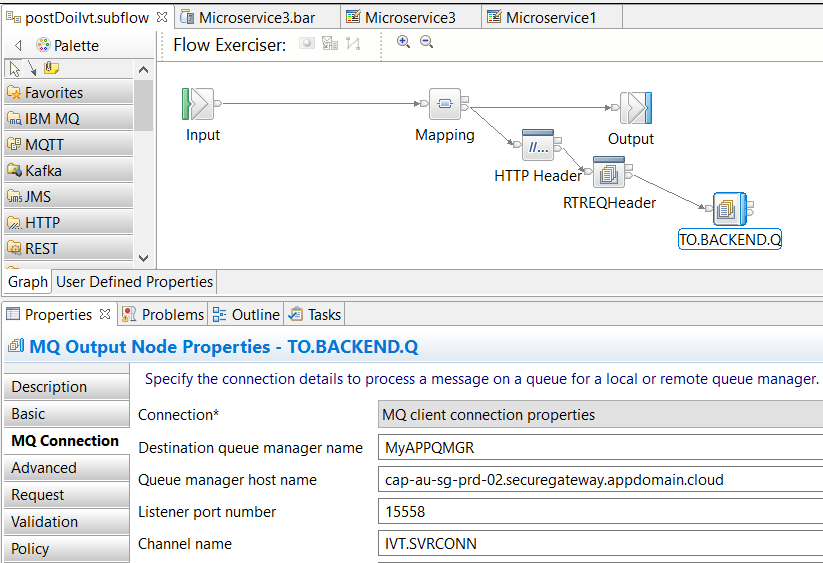
See document: ”1.Developer Experience for ACE MQ with RHOS Tools and Tekton v1.1”

Section: ACE MS 3 – REST Service exposing off RH OpenShift cluster MQ QMGR

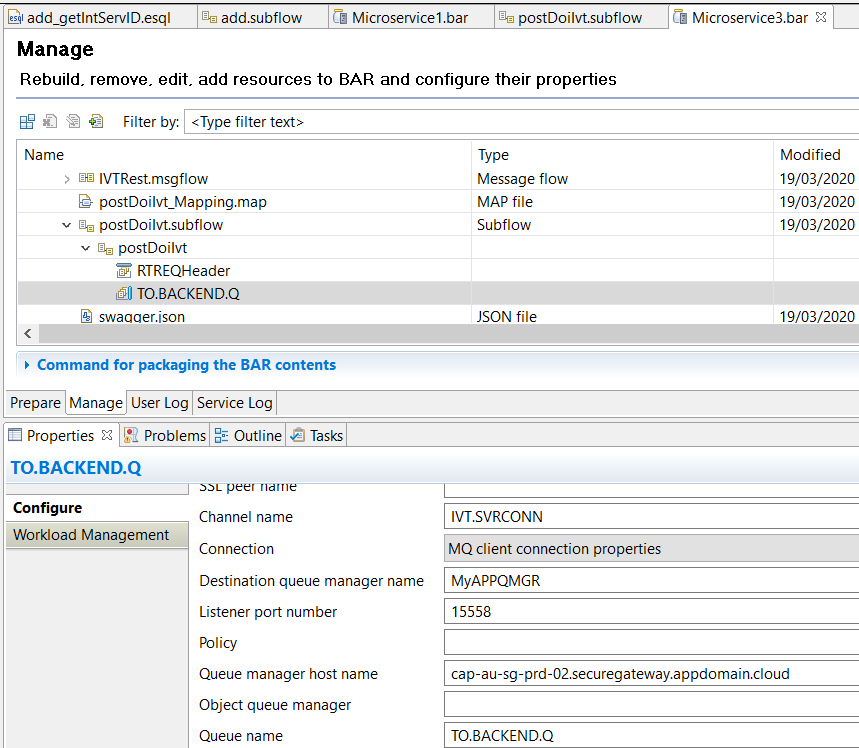
Page: 78

There are two options in the IBM App Connect Enterprise Toolkit for configuring the MQOutput node in ACE Microservice 3 to use the secure gateway connection to resolve connectivity to the MyAPPQMGR on your machine/laptop.

#### Option 1:MQ configuration update in the source message flow



#### Option 2:MQ configuration update in the BAR override



With the change made and the BAR file updated and saved you will need to:

1. Push the BAR to GitHub
2. Webhook trigger or manual RH OpenShift Console rebuild
3. Wait on successful build and deployment before continuing

These are the same steps you performed against ACE MS 1 in the earlier scenario in this article/document when you change ACE MS1 RESTRequest Node URL to point to the ACE MS2 stub.

## Testing MQ via IBM Public IP – using RFHUTILC (MQ client test)

RFHUTIL is a popular MQ test tool shipped as part of supportpac MS03.

RFHUTILC is the client version.

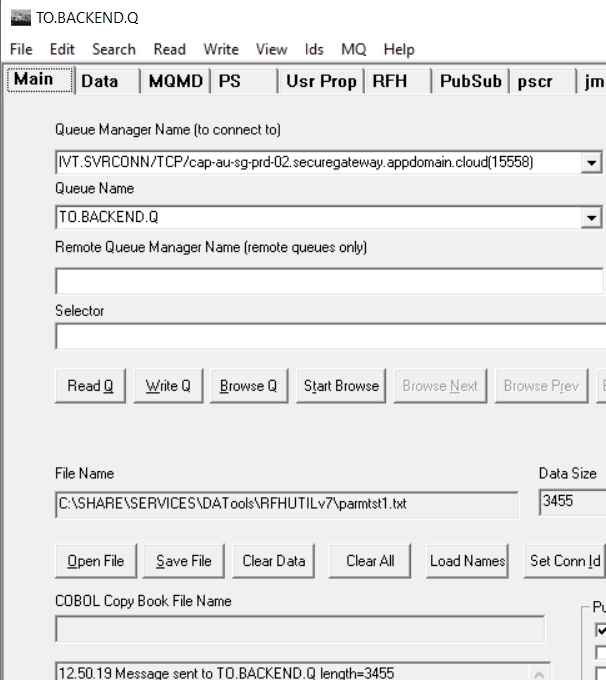
To connect to the queue manager MyAPPQMGR use the public IP address from the IBM Secure Gateway (copied earlier) as part of the connection URL for the IVT.SVRCONN channel in the following format in the “Queue Manager Name” parameter of RFHUTILC.

IVT.SVRCONN/TCP/cap-au-sg-prd-02.securegateway.appdomain.cloud(15558)

Set the Queue name to TO.BACKEND.Q

Open a file with some data to act as the message

Hit writeQ



Check the secure gateway client side logs via URL:

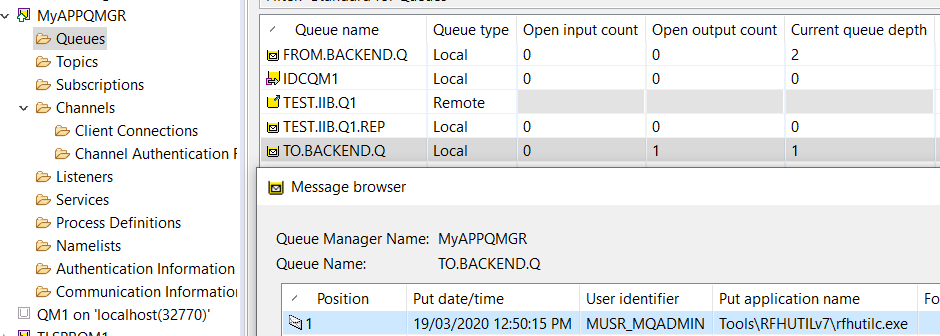
<http://localhost:9003/logs>

You should see RFHUTILs connection to the MyAPPQMGR via the public IP address.



### Check the result on the target queue manager

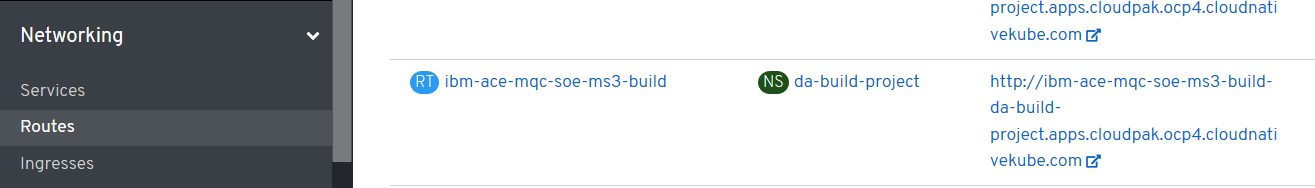
There should be a message on the TO.BACKEND.Q queue



## Testing ACE Microservice 3 to Put to MQ via IBM Public IP – Using a REST Client

### Obtain the RH OpenShift Route for ACE MS3

RH OpenShift Console->Networking->Routes->ibm-ace-mqc-soe-ms3-build



Or use the command:

Oc get routes



### URL and test data for calling ACE MS3 service

Add the ACE MS 3 base URL to the ipaddress of ACE MS3 retrieved from the route.

Using a REST Client POST to URL

<http://ibm-ace-mqc-soe-ms3-build-da-build-project.apps.cloudpak.ocp4.cloudnativekube.com/ivtrest/v1/doiIvt>

using data:

{

"name":"anyname",

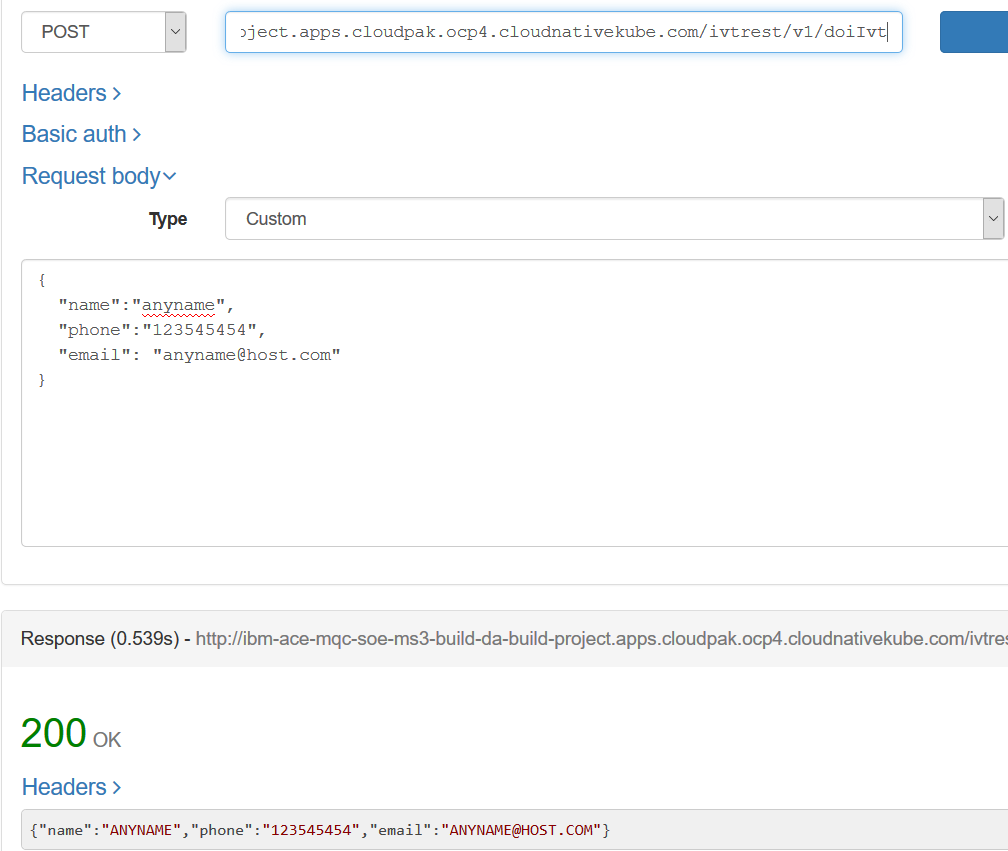
"phone":"123545454",

"email": "anyname@host.com"

}

You should receive an “Echo” style response from ACE MS3 with the data UPPERCASED

### Test with REST Client

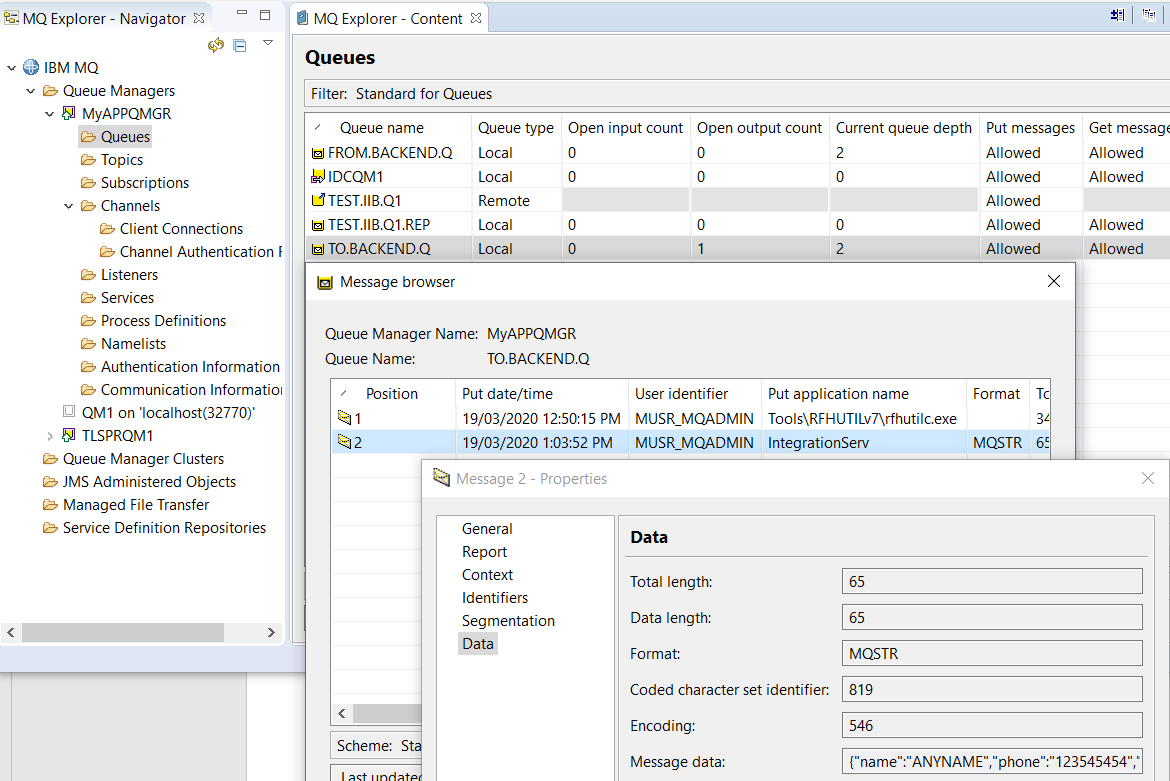


### Check Test Results on MQ

On Queue Manager: MyAPPQMGR

On Queue: TO.BACKEND.Q

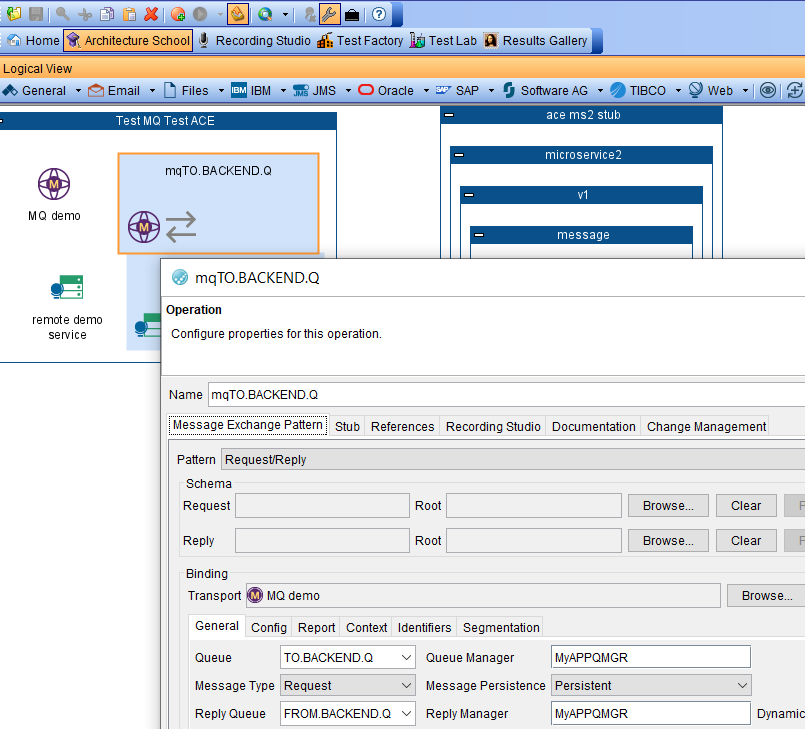
Data: JSON payload with the values UPPERCASED



## Using the MQ Stub on IBM Rational Integration Tester as a backend application

### Review stub parameters in IBM Rational Integration Tester

Architecture School->Logical View->Test MQ Test ACE->mqTo.BACKEND.Q



note that the IBM (Rational) Integration Tester stub is set up to Get from TO.BACKEND.Q and Put to FROM.BACKEND.Q.

Review the activities that the MQ Stub will perform when a message is received.

Test Factory->Logical->Test MQ Test ACE->mqTO.BACKEND.Q->Stubs->MQDemo

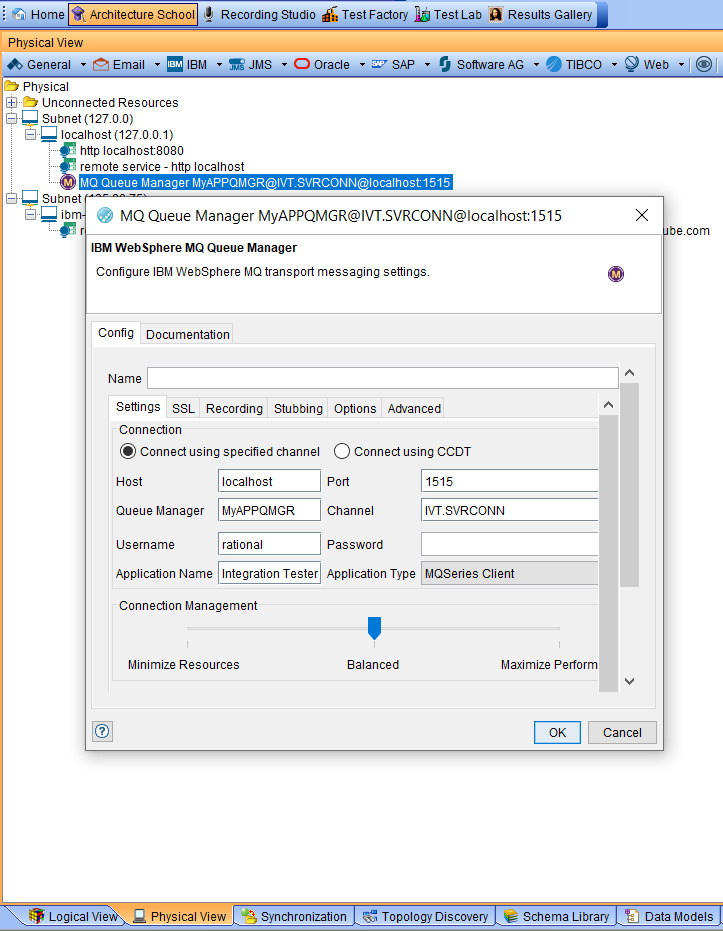
Double click on MQDemo and select Activity

Depending on the inbound data payload the stub will either append the message “This is from MQ stub” to the original message or just place “This is from MQ stub” on FROM.BACKEND.Q



Finally review the Physical view to check the connection details that the IBM Integration Tester will use for the Stub to connect to MQ.

Architecture School->Physical View->subnet localhost->MQ Queue Manager MyAPPQMGR@IVT.SVRCONN@localhost:1515





### Stub uses IVT.SVRCONN channel definition in the MyAPPQMGR

The IBM (Rational) Integration Tester stub will use a SVRCONN channel to connect to the MyAPPQMGR. The channel definition is as follows:

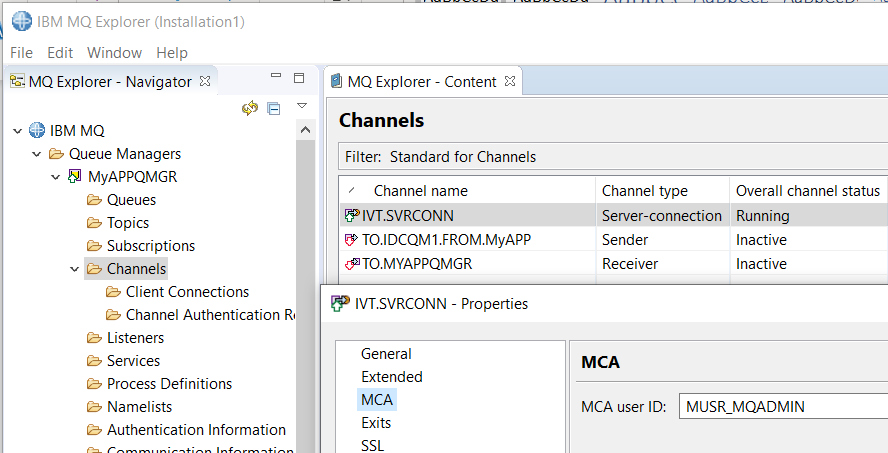
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN) REPLACE

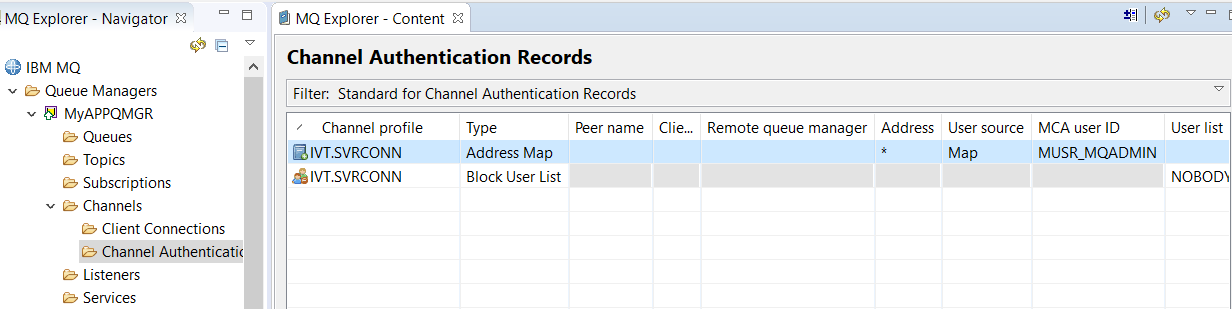
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)

SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(\*) MCAUSER('')

REFRESH SECURITY TYPE(CONNAUTH)

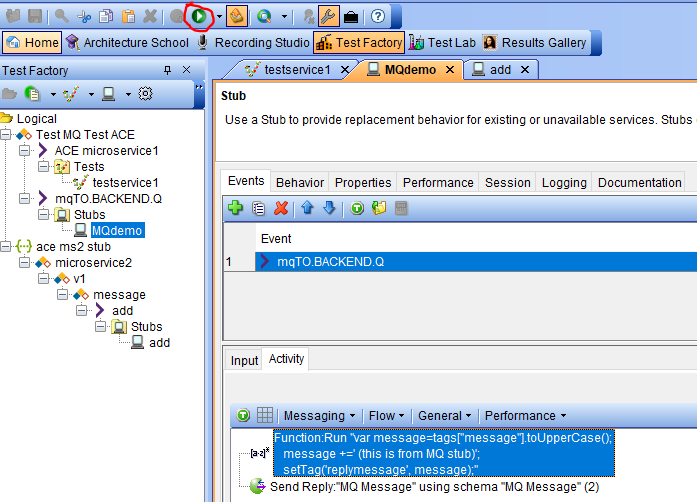




### Start IBM Rational Integration Tester Stub to retrieve messages placed on MQ by ACE MS3

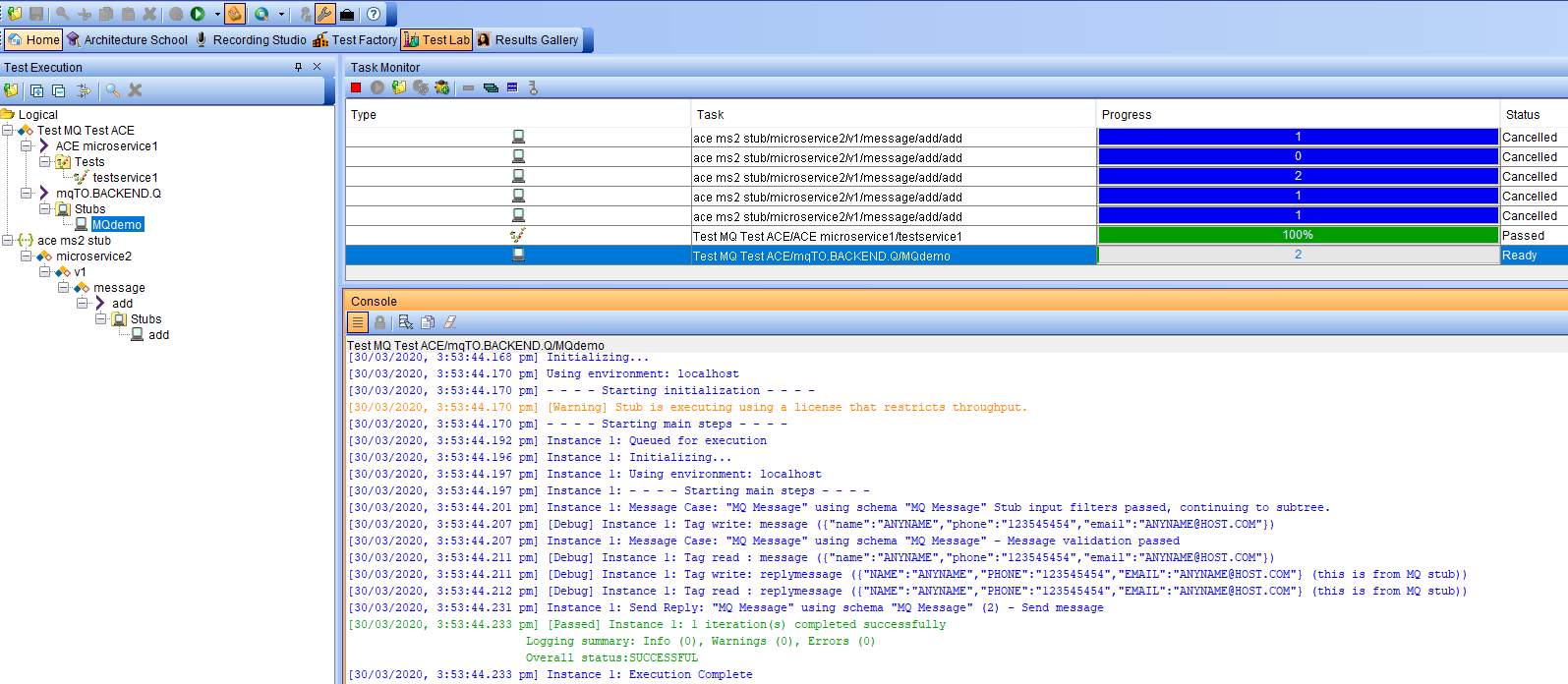
In the Test Factory expand mqTo.BACKEND.Q->Stubs->MQDemo

Select the MQDemo stub and hit the green “play” button to run the stub



### IBM Rational Integration Tester Stub receives message from ACE MS3

In the Task monitor console in the Test Lab tab observe the Stub reading and writing messages successfully



### Review MyAPPQMGR queue FROM.BACKEND.Q contents

Depending on the test you have run with RHFUTIL and the actual ACE Micro service 3 you will have a mixture of messages on the FROM.BACKEND.Q

Type 1: (this is from MQ stub)

Type 2:{"name":"ANYNAME","phone":"123545454","email":"ANYNAME@HOST.COM"} (this is from MQ stub)

