MORC Integration Director (MINDI)

Modular OSGI Runtime Container (MORC) Framework :-composed of Three main components:

a) **Bridge Application:-** Runs the Felix OSGI container in an application server (e.g., WebSphere). Provides the bridge from application server HTTP threads to the OSGI container

b) **MINDI:-** MORC Integration Director :- User Interface for the dynamic configuration capability in MORC. Can be configured to connect to multiple environments. Access business web services in each environment to read and update configurations, no direct DB connection

c) **Core Framework Bundles :**- UHG properietary application build on OSGI bundles that provide base functionality: core, configuration services, unit test framework

Required for the Bridge and MINDI applications to function. Provides Business Framework base classes, interceptors, logging, and many other framework capabilities for service development

HOW TO ADD UPM MINDI Config URATION ENTRIES IN UPM 3 MINDI Config APP

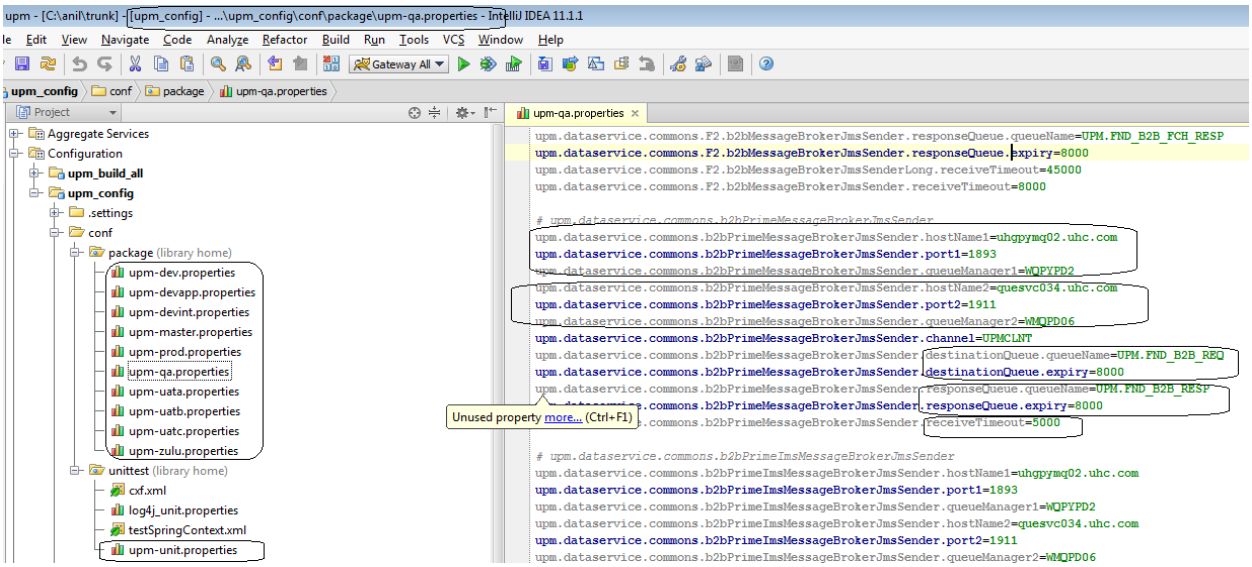
1. **Pre-requisite:** All AE developers who work on UPM services should request to add the following groups to their Windows id using SATS:

   UPM3_AE_TST : for access to UPM3 services in non-prod environments.

   UPM3_AE_PRD : for access to UPM3 services in production.

   Do raise secure request for getting access on **upmAdmin** for Windows platform to modify and add in this application

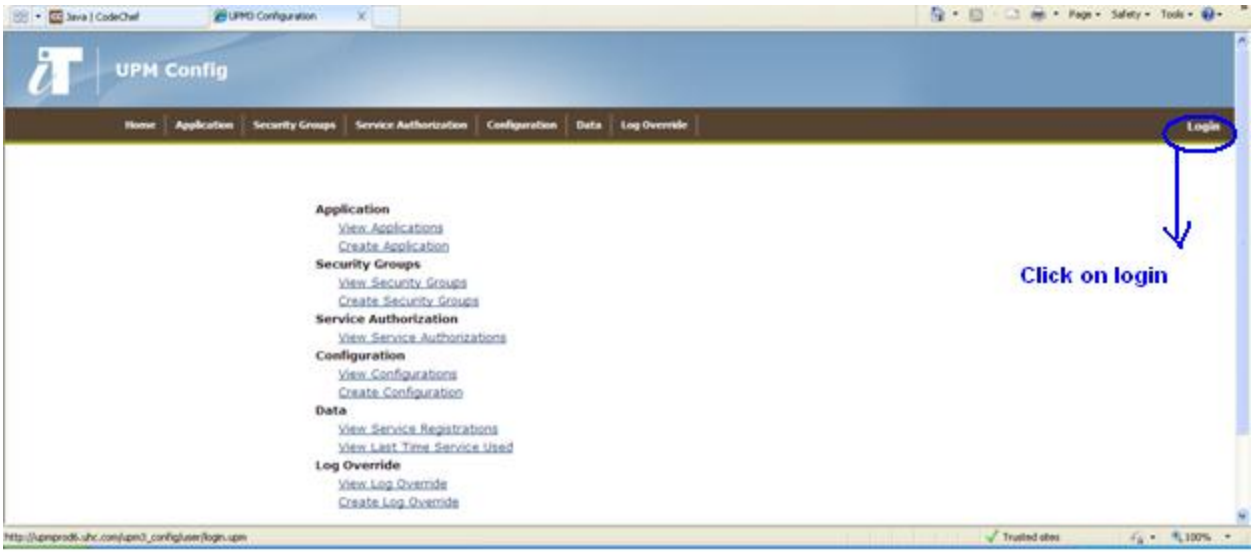
For UPM2 to UPM3 Migration:- In UPM2 we were adding service configuration in properties file for each respective non-prod and prod environment. UPM2 configuration properties for all environment exist under upm_config.



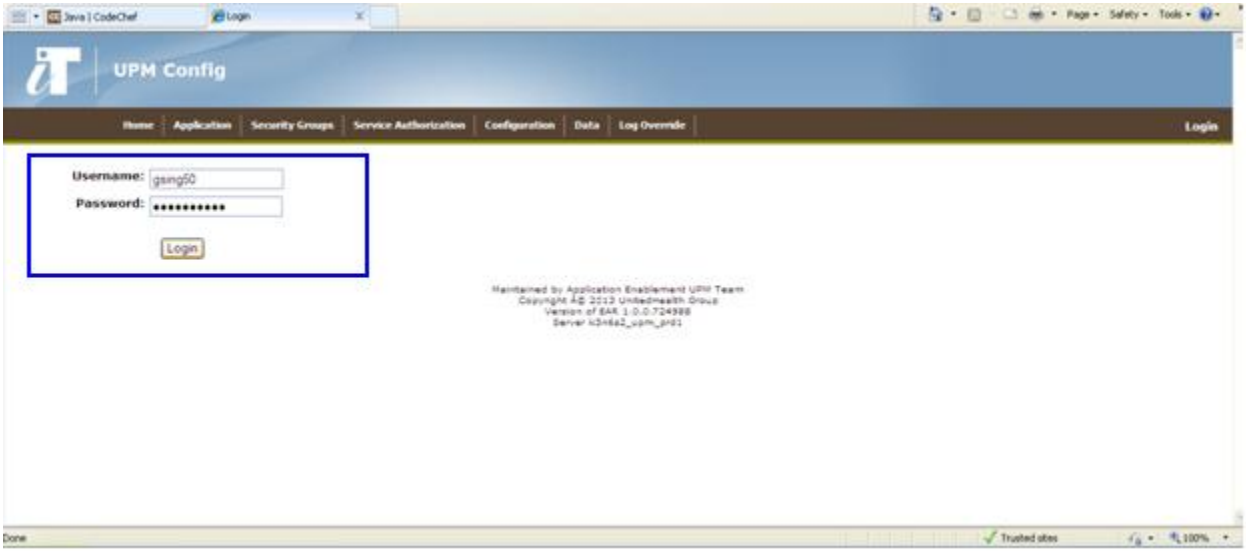
With UPM3 we moved configuration properties in database rather than referring from files. UPM3 MINDI Config app allows developer to add UPM3 service configuration into database. This application allows developer to add and edit configurations.

Each developer need to make sure the configuration values of upm3 service must match what it exist in upm2 service while migrating upm2 service to upm3, which means endpoint, timeout, queueManager etc. properties what we have in UPM2 for the service in must have same value in upm3 configuration app.

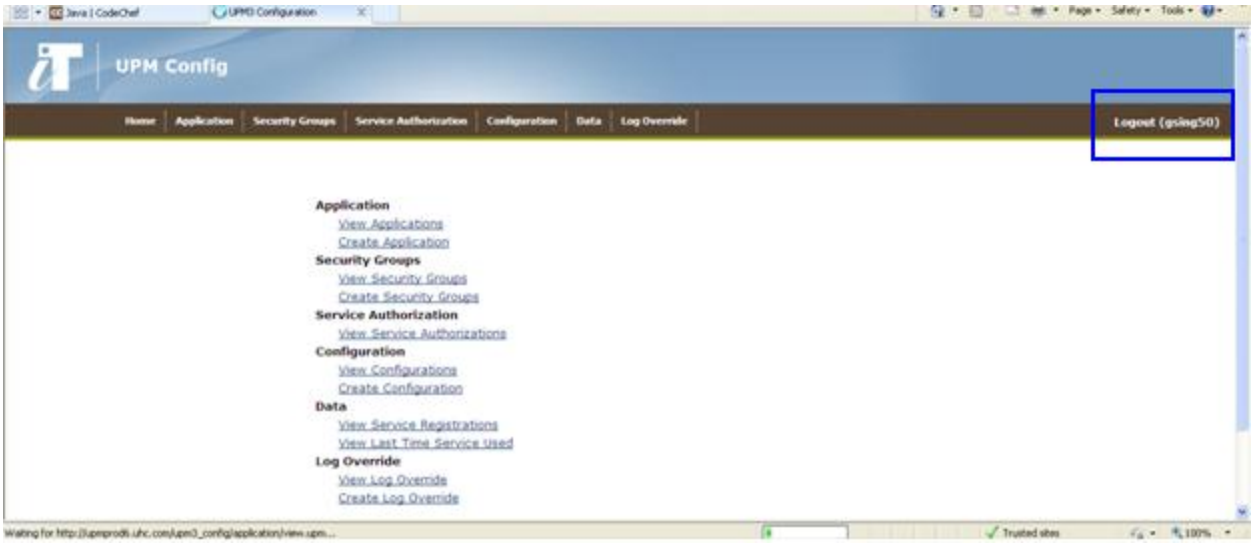
2. Go to internet browser and open this url:
   a) **Non-production environment**: http://u3cstage.uhc.com:81/mindi/ - to display/create/edit/update all non-production environment configuration.
   b) **Production environment:** http://u3c.uhc.com:81/mindi/ - to display/create/edit/update production environment configuration. Each developer after completing service development of their service, they should run their service pointing to prod environment and add service configurations in prod with respective production configurations.
3. You will get following home page and then click on **Login** link.



4. This will ask to enter your authentication details, please enter your **NT ID** and **password** and click **Login**. Only authorize users (with upmAdmin) are allowed to access and modify.



5. Once you are successfully logged in, you will see this home page and your user name on the right top side of browser.



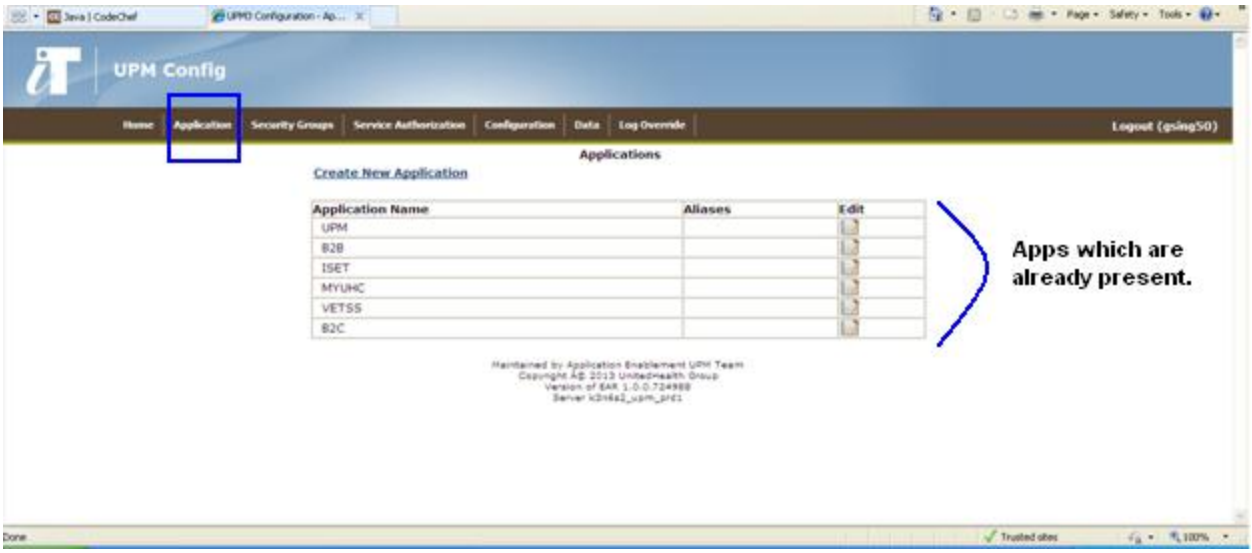
6. UPM 3 MINDI Configuration application setup process is divided into 6 steps:
    - Application Name setup.
    - Security Group setup.
    - Service Authorization setup.
    - Configuration setup.
    - Data
    - Log Override

Data functionality helps us to view registered services and last time service used by client application. And Log override to change log level for a particular service for a particular environment for defined duration.

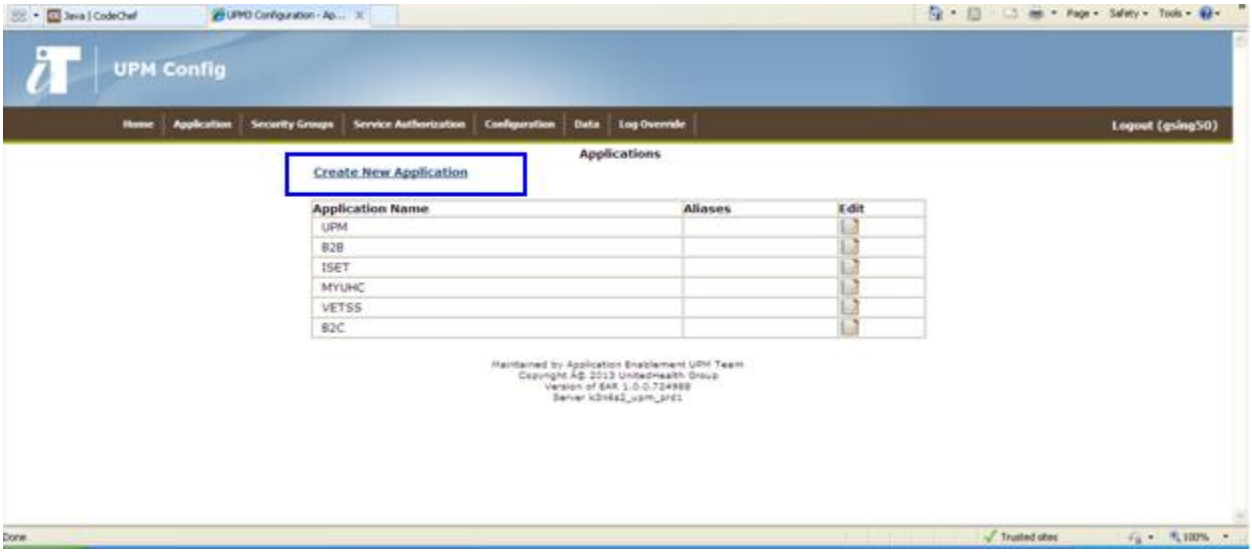
To add configuration for a service, we will be using first 4 functionalities. Let's start with 1st step,

1. **Application Name entry**:-

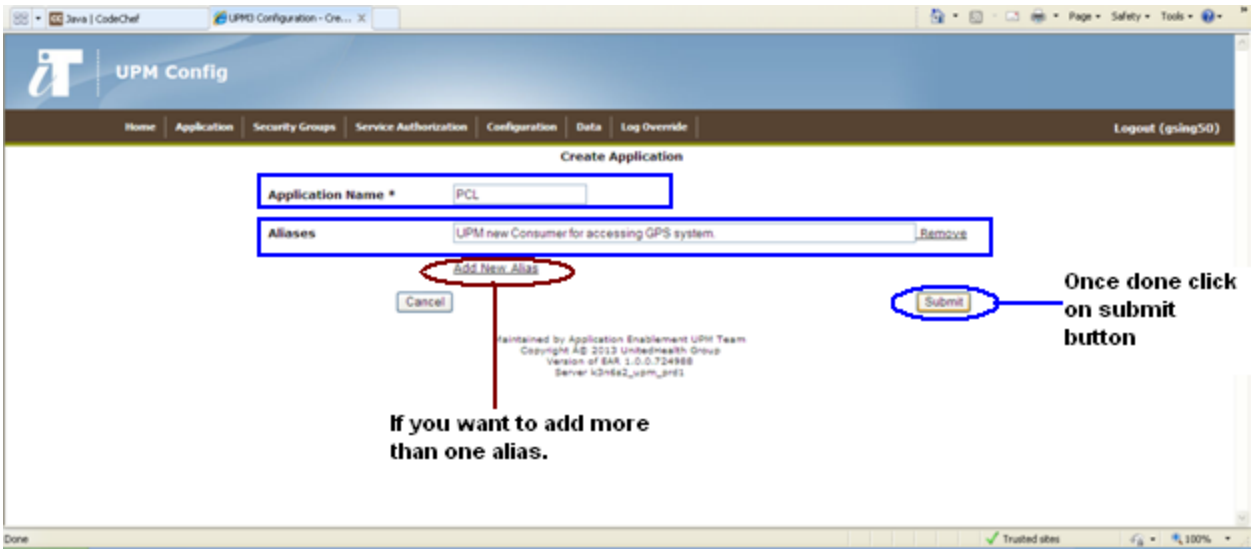
    a. Click on **Application** link and select **View Applications** which will show you all the applications which are already added to this UPM 3 configuration app. Displayed Application Name will exist in all non-production environments.



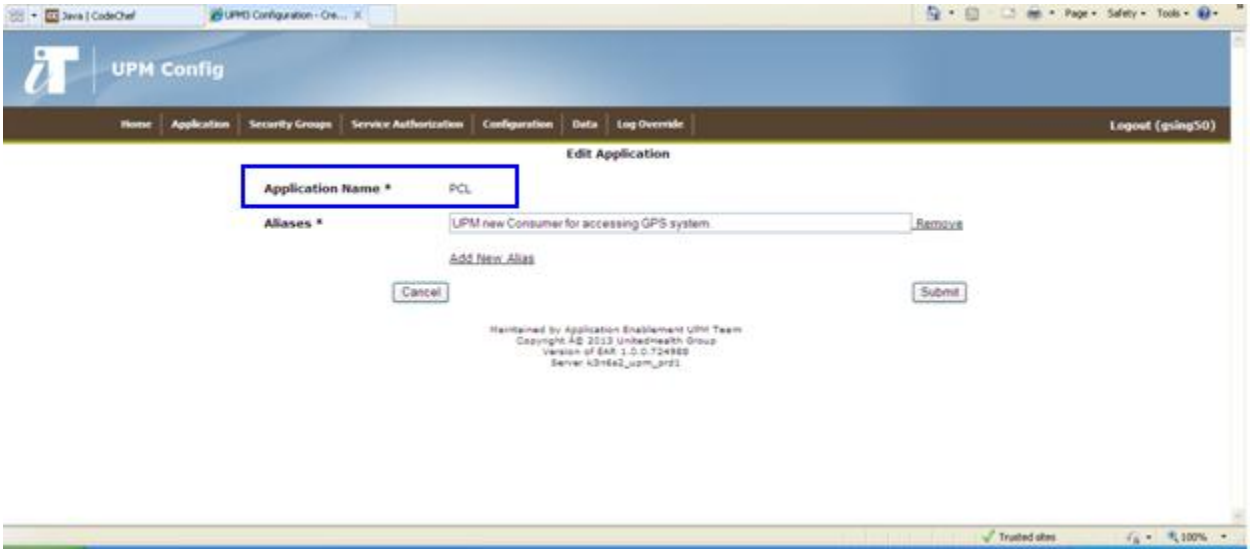
b. If application name which you want to add is already present, then move to next step. Otherwise, click on link **Create New Application** as shown in figure below:



Then, you will see this page where you will add application name is all uppercase in **Application Name** field and his alias name if any or general description about that application in the **Aliases** field.

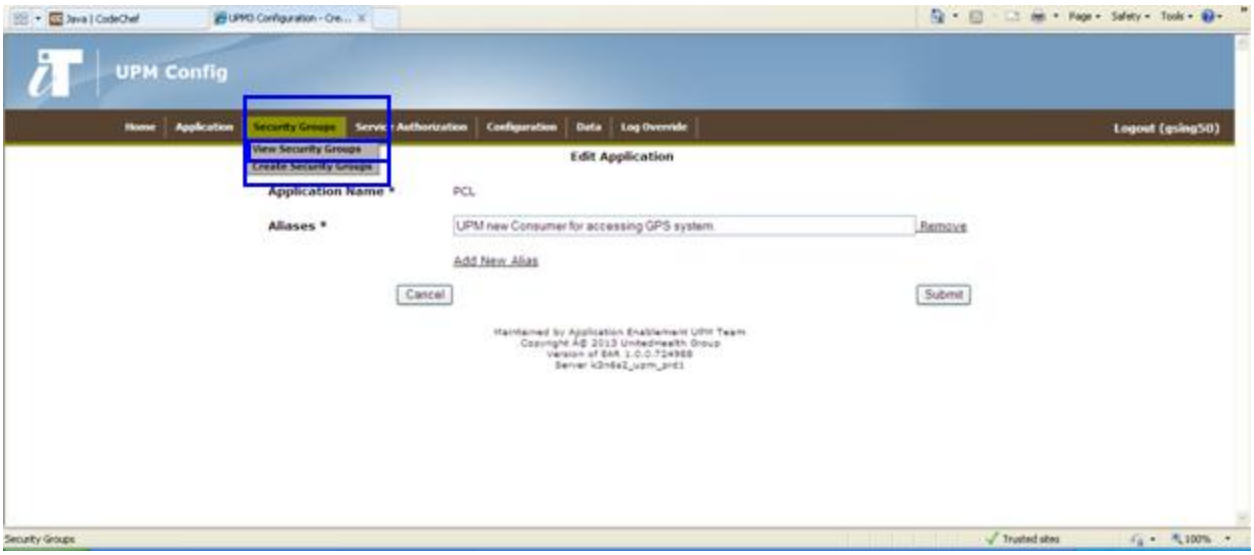


c. Once done click on **Submit** button and your changes will be added. Now you will see the same screen with your application name as non-editable field showing that your entires have been added in all non-production enevironments.

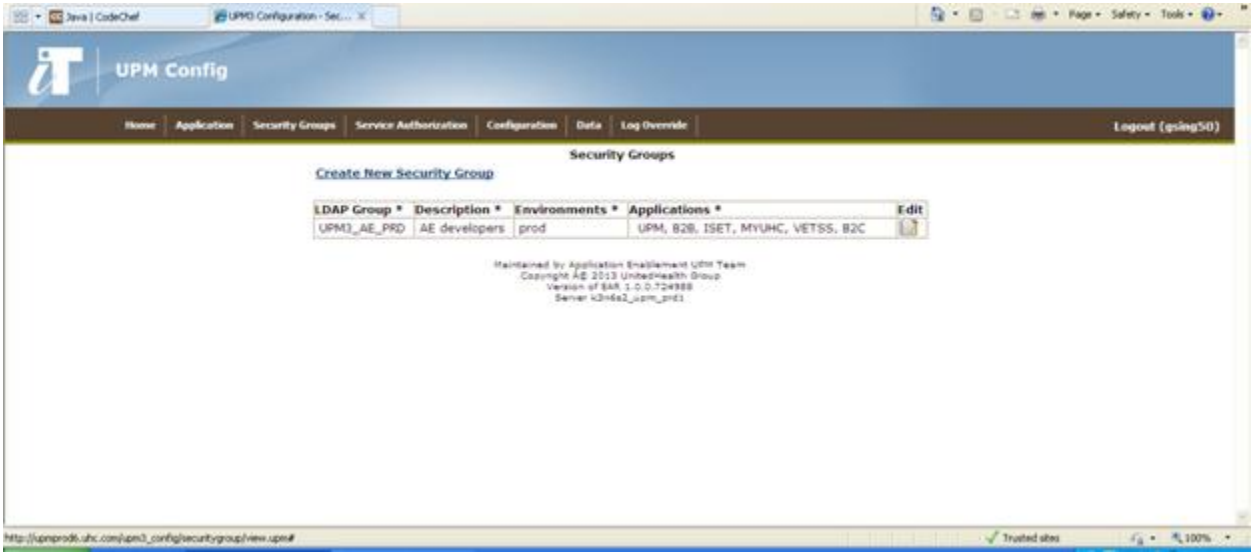


2. Configuring Security Groups:-

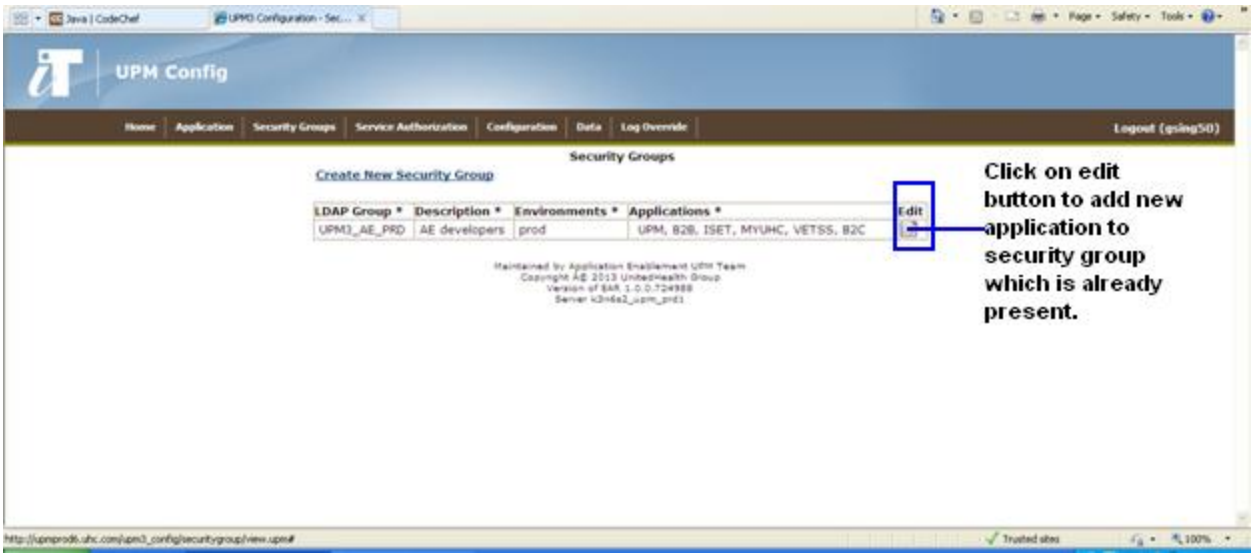
a. Now next step is to add LDAP Security Group of UPM client/consumer application. To view all existing LDAP security Groups. Click on **Security Groups** link and select **View Security Groups**.



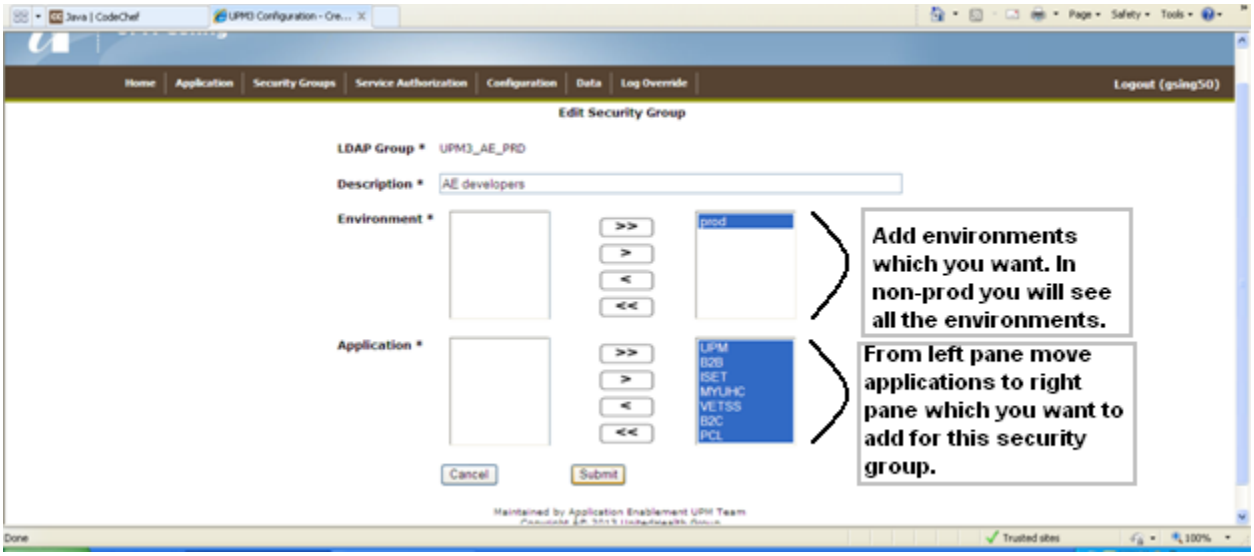
You will see all the LDAP security groups which are already added to UPM 3 config app for UPM and its clients in that particular environment.



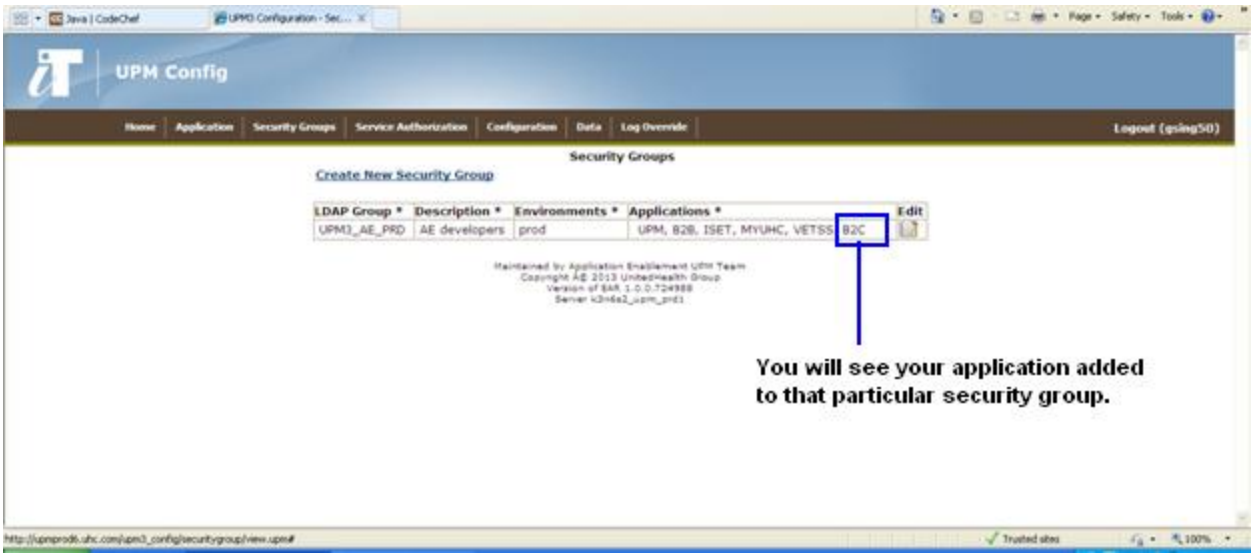
b. Now, if the application security group is already present, you will need to add your application into that group by clicking on **Edit** button.



Next page will open selected security group in editable mode.



c. Once done, then click on the **Submit** button to save your changes. Now you will see your application added to that particular security group.



***Note:** For UPM developers, all applications should be also added to **UPM3_AE_TST** security group for non-prod environment configuration and **UPM3_AE_PRD** security group for production environment.

For non-prod environments:

UPM3_AE_TST	Access for AE developers	master, charlie, zulu, unit, dev, bravo, stage, alpha, systest	UPM, MYUHC, OMX, B2B, VETSS, ISET, ILEAD, RXHD, IIM, TRICAREONE, UHCMW, TWBS, TWPS, B2C, HAZ, DocGPSMW, LOADTEST, PCL	

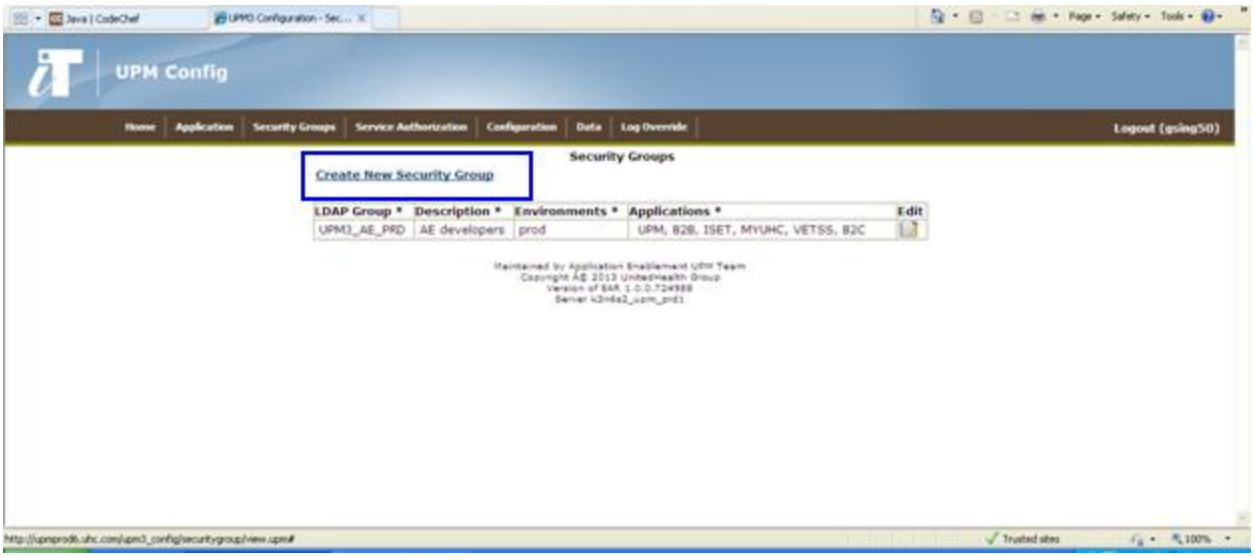
For prod environment:

LDAP Group *	Description *	Environments *	Applications *	Edit
UPM3_AE_PRD	AE developers	prod	UPM, B2B, ISET, MYUHC, VETSS, B2C	

Note :- Every client application will be creating their LDAP security group (which will be something like UPM3_XXX_TST or PRD) to access UPM3 services. Client users will be raising secure request to be part of this LDAP security group, which will be managed by Client. At UPM layer we will be authorizing client LADP security for particular environment/s.

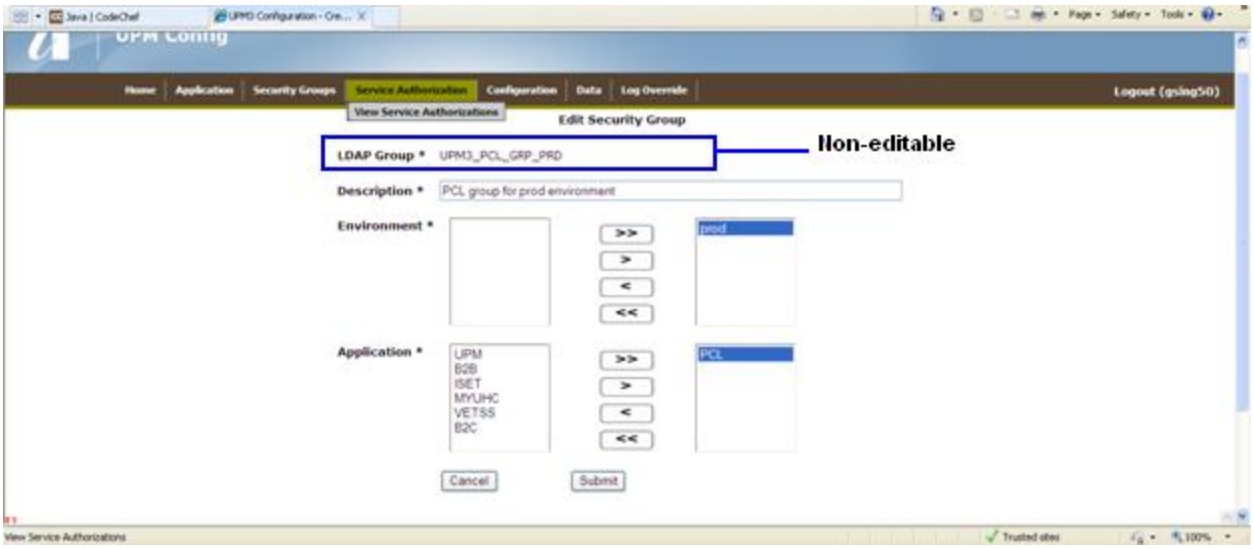
Now if security group is not already present, add new security group by clicking on **Create New Security Group.**

Note :- Consumer will inform our SDM about the name of LDAP security group which belongs to them. Then SDM inform Dev Manager/Lead for the need of creating and configuring consumer specific LDAP security group, which may get down to Developer to set up same in UPM3 config app.

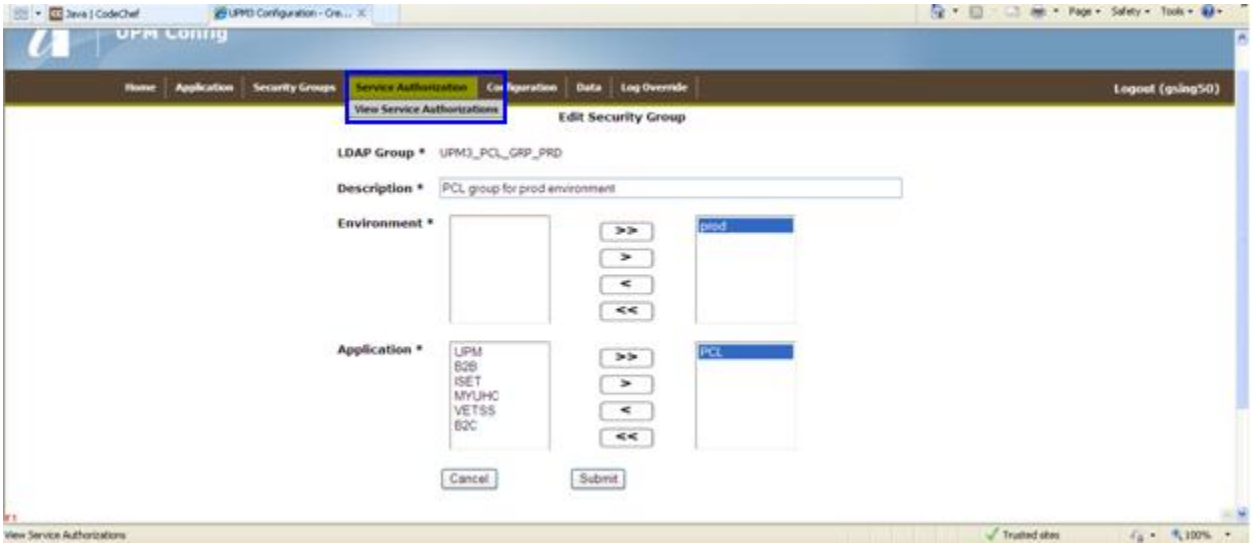


Add all required information for new security group as shown in figure below:

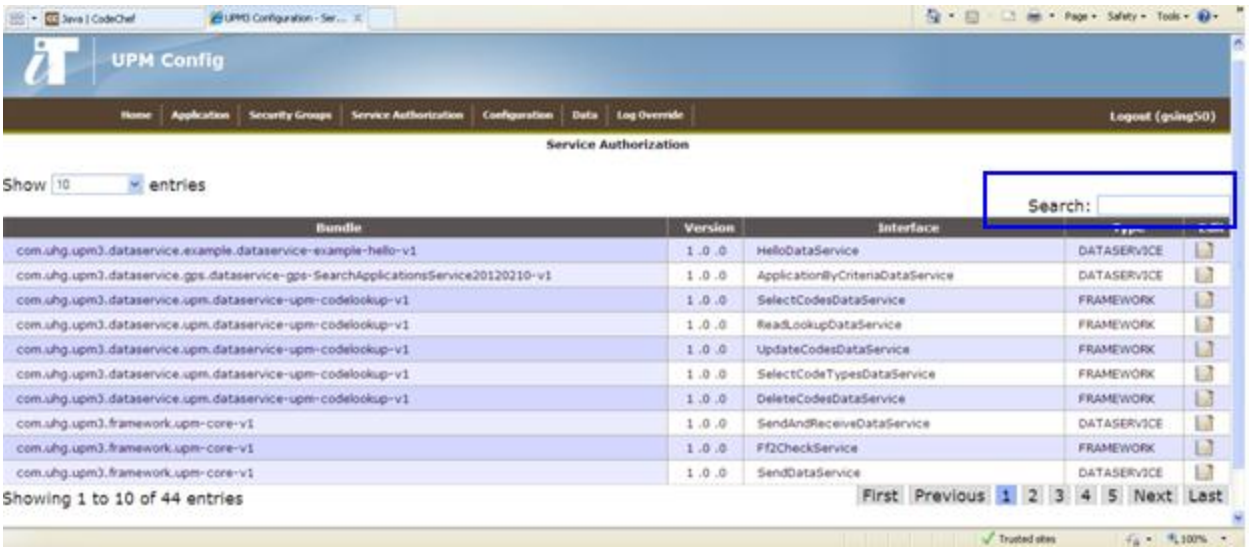
And click ion **Submit** button to save all changes and you will come to this screen with LDAP group as non-editable field confirming your entries for security group has been saved.



3. **Service Authorization: -** Next step is to add service authorization entry, basically to authorize your **aggregate service** bundles for the new consumer (application names). So, that new consumer which you have added in previous step should be authorized for the service bundles. To do this click on **Service Authorization** Link and
    a. Select **View Service Authorizations** as shown in picture below:

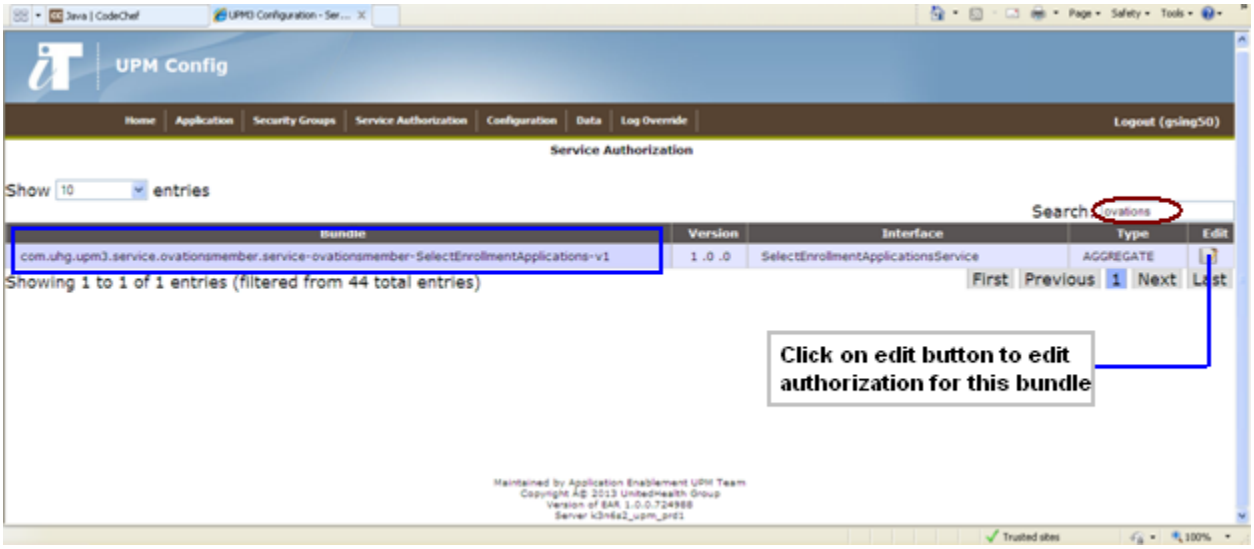


You will see all the bundles in this page and you can search your **aggregate service** bundle by typing in your entry in the **Search** field.

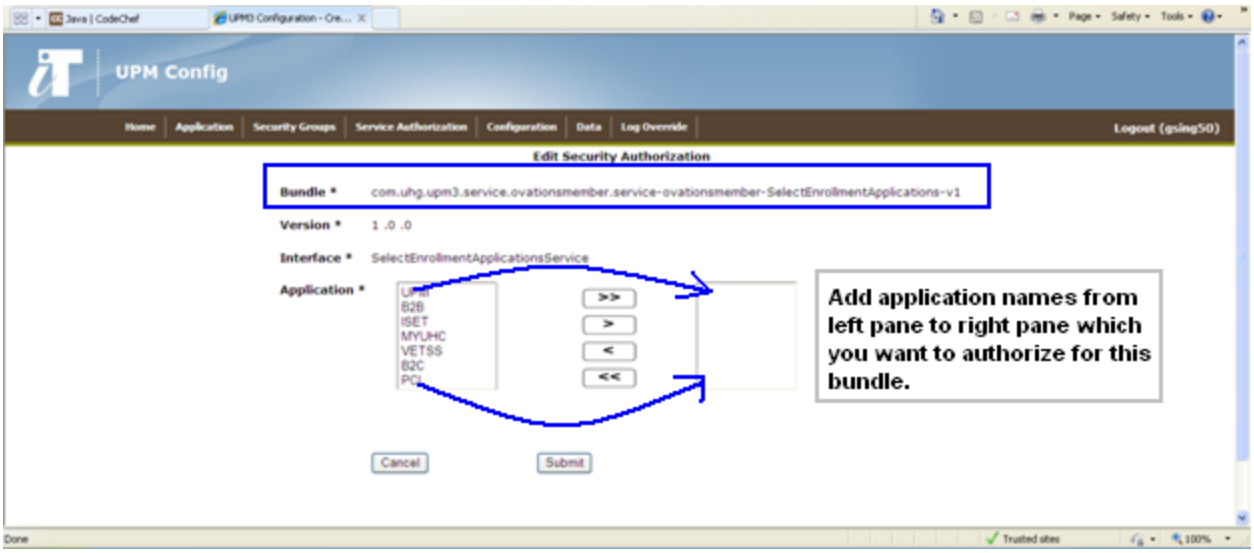


For example if your aggregate service contains word **ovations,** you need to type ovations word in search box and it will show you all the bundles containing this word. By clicking on edit button corresponding to that bundle you will be able to edit the service authorization configuration.

Note: - With our UPM development guidelines, all newly developed Service/Aggregate or Data Service Bundles should be suffixed by version # i.e Interface column should be SelectEnrollmentApplicationsServiceV1

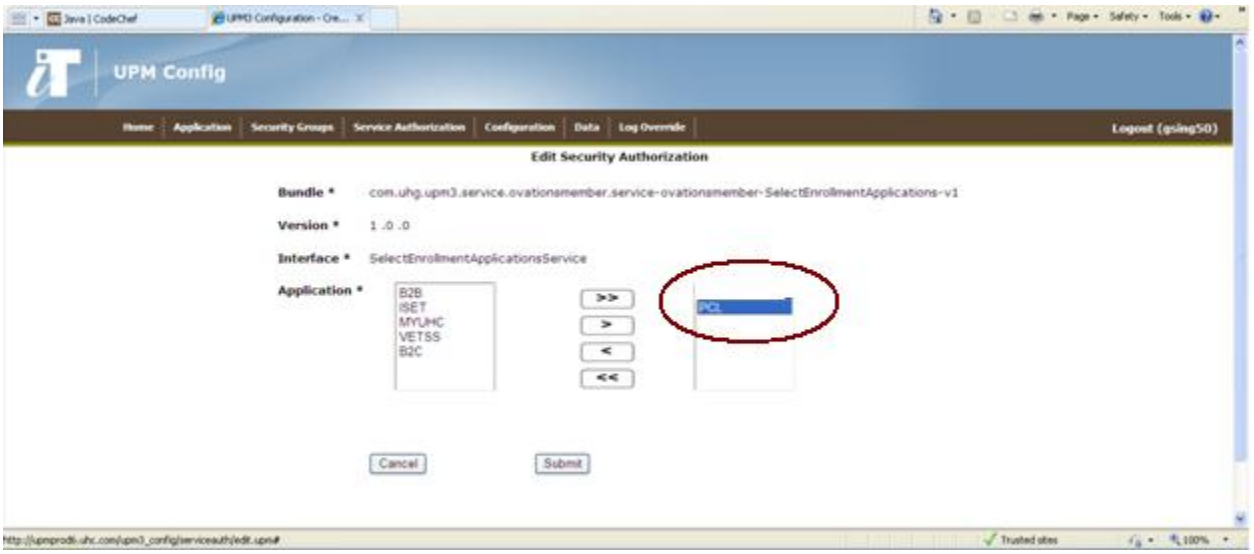


b. In next editable page you will see below mentioned screen:



Add all the application names which you want to authorize for this selected bundle by selecting name in left page and clicking on arrow button.

Then click on Submit button to save your service authorization changes.



Same steps you need to follow for all dependent **aggregate** bundles.

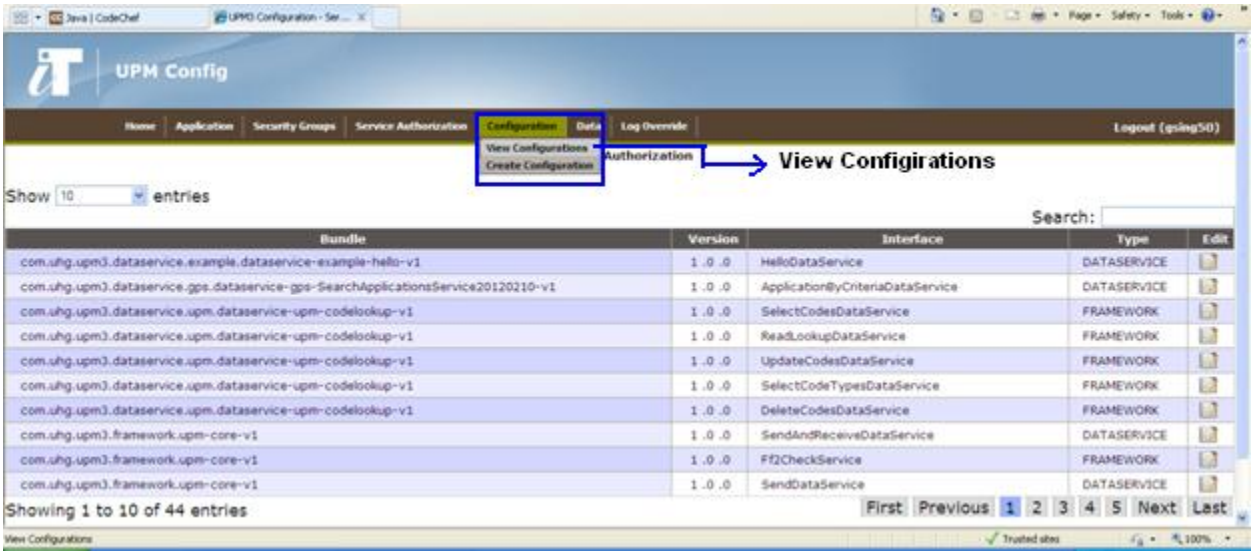
Note: - Service and data service bundle will get register and automatically visible under Service Authorization, when developer run Junit test case or invoke service. e.g.

i) When developer publish wsdl then service bundle get registered automatically and visible under service authorization.

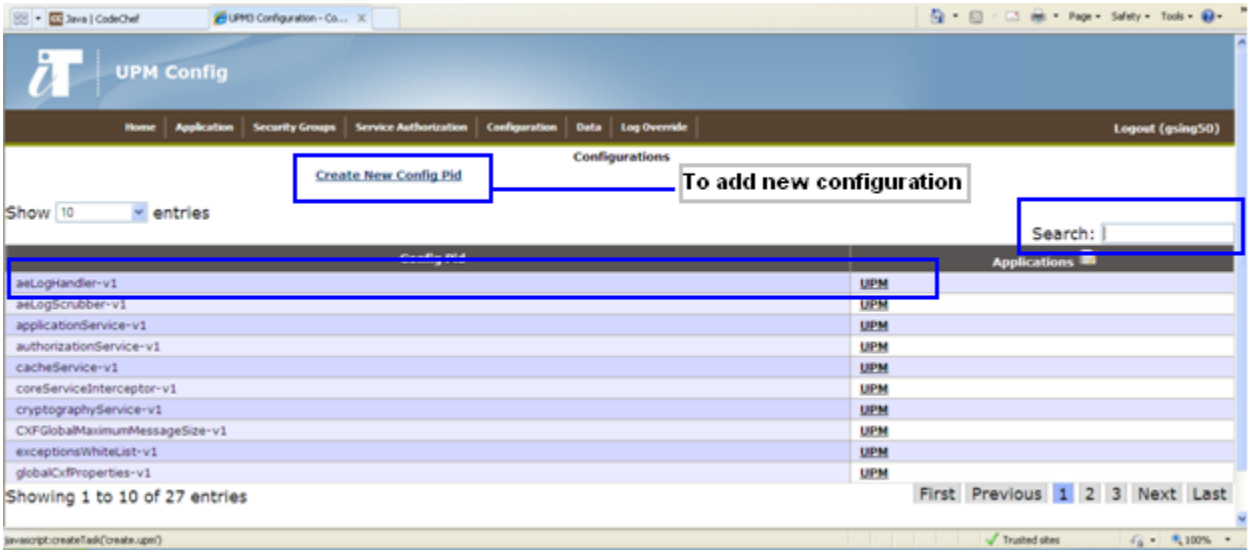
ii) When developer run Junit data service test case then data service bundle will get registered under service authorization and when developer run Junit service test case then it register service and data service bundles.

iii) When we run service from soapui then both service and data service bundle will get registered under service authorization.

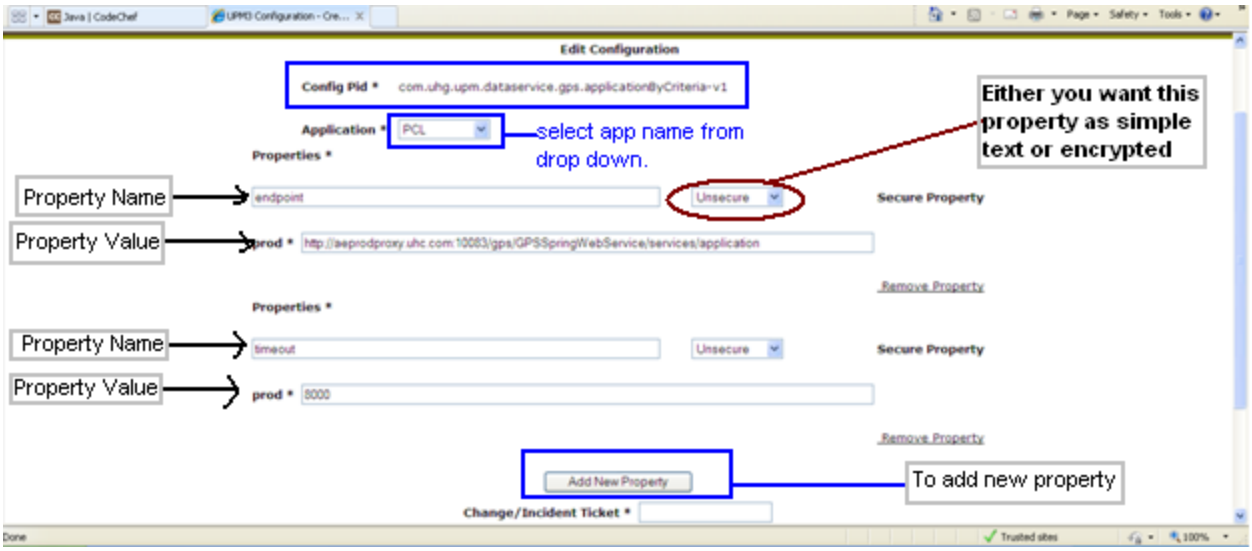
4. **Configuration: -** Last step is to add you property configurations. To do that click on **Configuration** link and select **View Configurations.**



This will show you all the existing property configurations:



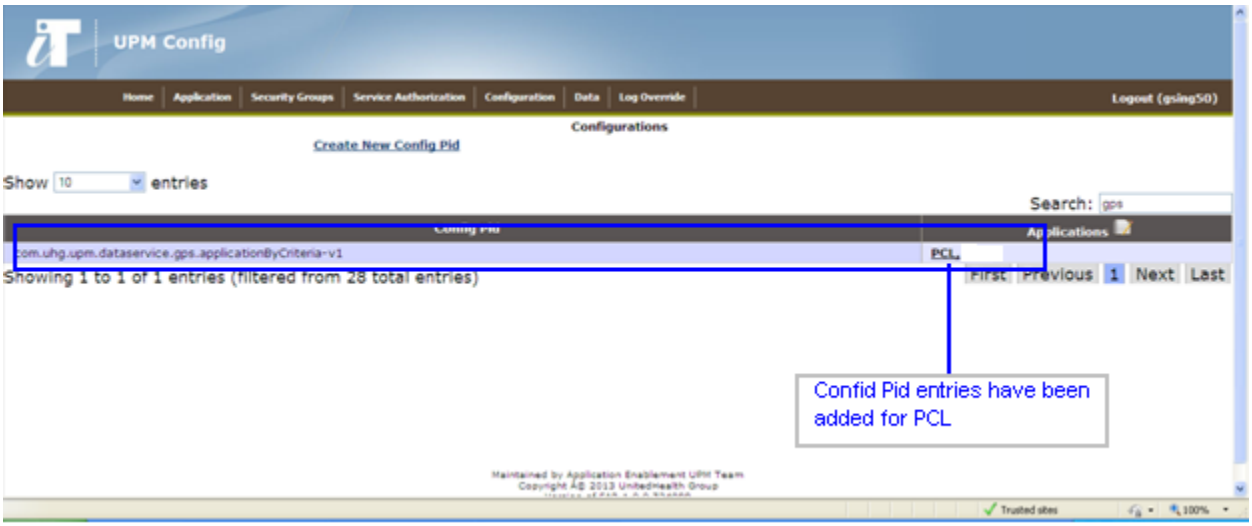
Property configurations are based upon **PID** which you have provided in your spring context file in your module. First search for your PID if that already exists, you can add new properties which you want to add, otherwise click on Create **New Config Pid** to add new config Pid with properties.



Fill in all the required details as shown in screen above corresponding to that PID.

If you want to **copy same configurations for some other application**, use the drop down box on the lower part to page to copy all the configurations to other selected application.

Once done, click on **Submit** button and your changes will be saved.



Note: Sometimes UPM3 MINDI Config app takes more time to add entries to data base and it may happen that you will not see you entries immediately, so try after few minutes and you will be able to get your entries. Do not add entries multiple times if you it won't show up immediately.

UPM service call either web services based backend or legacy backend (mainframe) through MQ/WMB. UPM3 MINDI Config app allow you create configurations for both backends

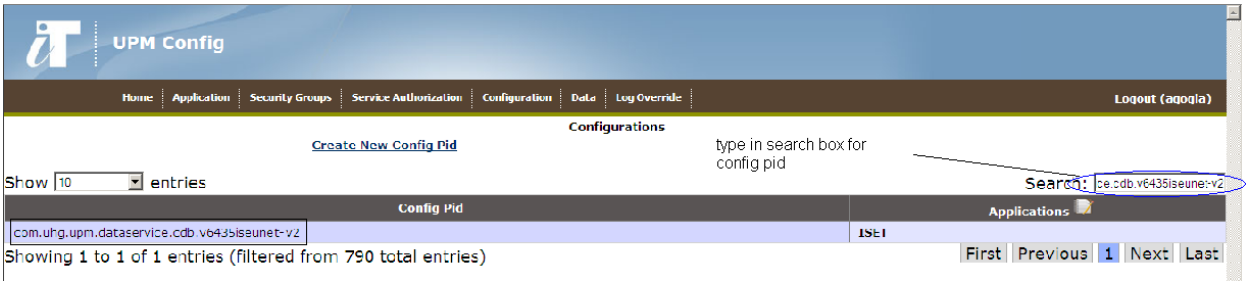
I. UPM3 MINDI Configuration for WMB based services :-

To add upm3 configuration for services calling legacy/mainframe based backend via WMB and FF, we need to create configuration pid (persistent id). Configuration pid consists of properties. Property (key-value) can have value either as textual or another reference of configuration pid(secondary config pid).

For wmb based configuration pid have **two basic properties** (key-value pair) i.e.
i) callingApplication property and ii) jmsSenderConfigBeanPid property.

Note: - we can have more properties apart from above two basic properties mentioned above. By default UPM3 framework provide basic default governorPoolPid. But with UPM3 config app you can provide custom governorPoolPid.

First check for your service config pid, to make sure it doesn't exist. Type your service config pid in search text box. If do exist then verify that it is created with all config properties.



Config pid name must match with one mentioned in spring xmls under resources/Meta-INF/spring of dataservice module. Naming convention we follow for config pid is to be in lowercase.

Add two properties for data service config pid :-

i) **callingApplication** property :- mention consumer application name for which you want to enable your service. Main config pid must be enabled for consumer application and secondary config pid must be enabled for "UPM" rather than for any specific conuemer

application as main config pid will use/refer secondary pid internaly.

ii) **jmsSenderConfigBeanPid** property :- jmsSender configuration property refer another config pid(secondary config pid) , which have further five properties, which will be :-

a) destinationQueuePid :- will have jmssender configuration information what we have in upm2 services under upm_config/conf/package/upm_xxx.properties
b) primaryJmsTemplatePid
c) secondaryJmsTemplatePid
d) responseQueuePid
e) setCorrelationId

Remove Property

Properties *

jmsSenderConfigBeanPid | Unsecure | **Secure Property**

unit * jmsSender-ff1-ISET-v1

dev * jmsSender-ff1-ISET-v1

systest * jmsSender-ff1-ISET-v1

alpha * jmsSender-ff1-ISET-v1

bravo * jmsSender-ff1-ISET-v1

charlie * jmsSender-ff1-ISET-v1

master * jmsSender-ff1-ISET-v1

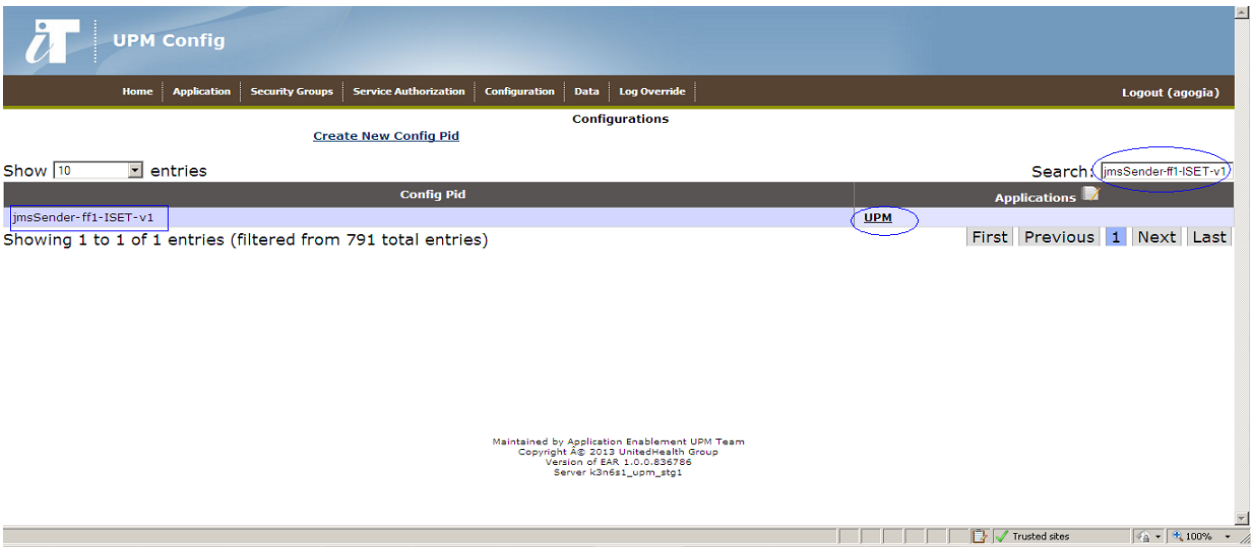
zulu * jmsSender-ff1-ISET-v1

stage * jmsSender-ff1-ISET-v1

Remove Property

Note: - properties name are case sensitive. Please use same properties name what you are seeing, as our framework will be using these properties name.

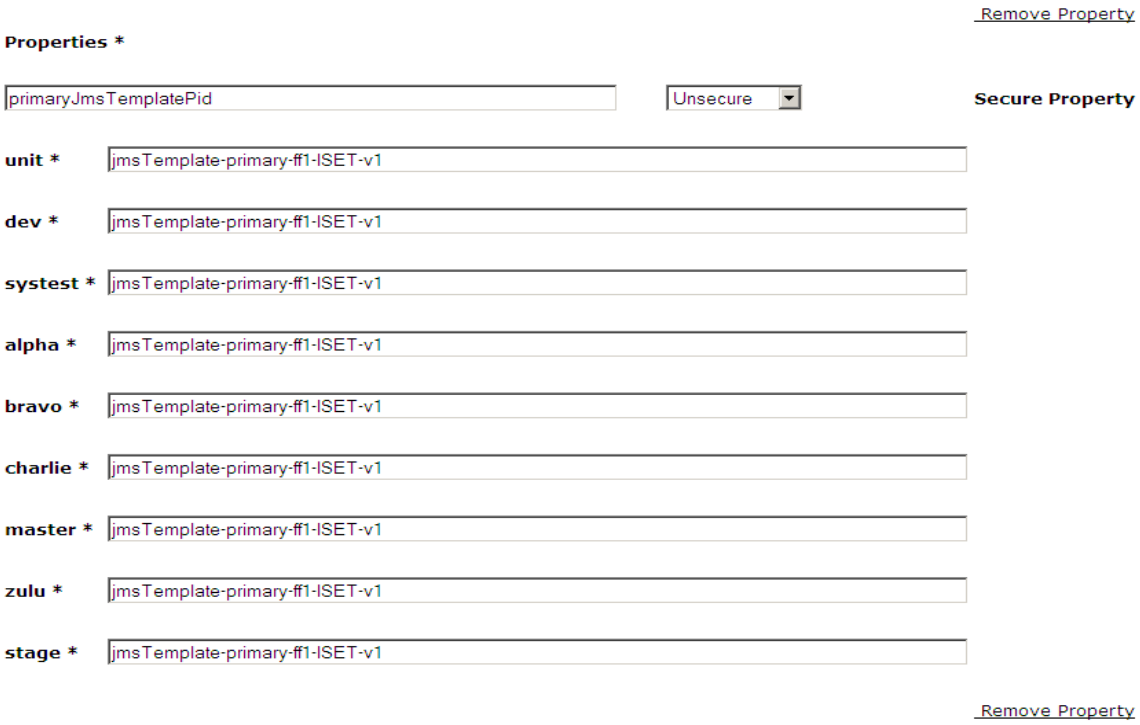
jmsSenderConfigBeanPid is another configuration pid which can further have their properties. Search for jmsSenderConfigBeanPid by going to view configuration screen and create this for “UPM”, since its not specific for any client and will be used internally for dataservice configuration pid for upm service. So we will enable this config pid for “UPM” in Application drop down.



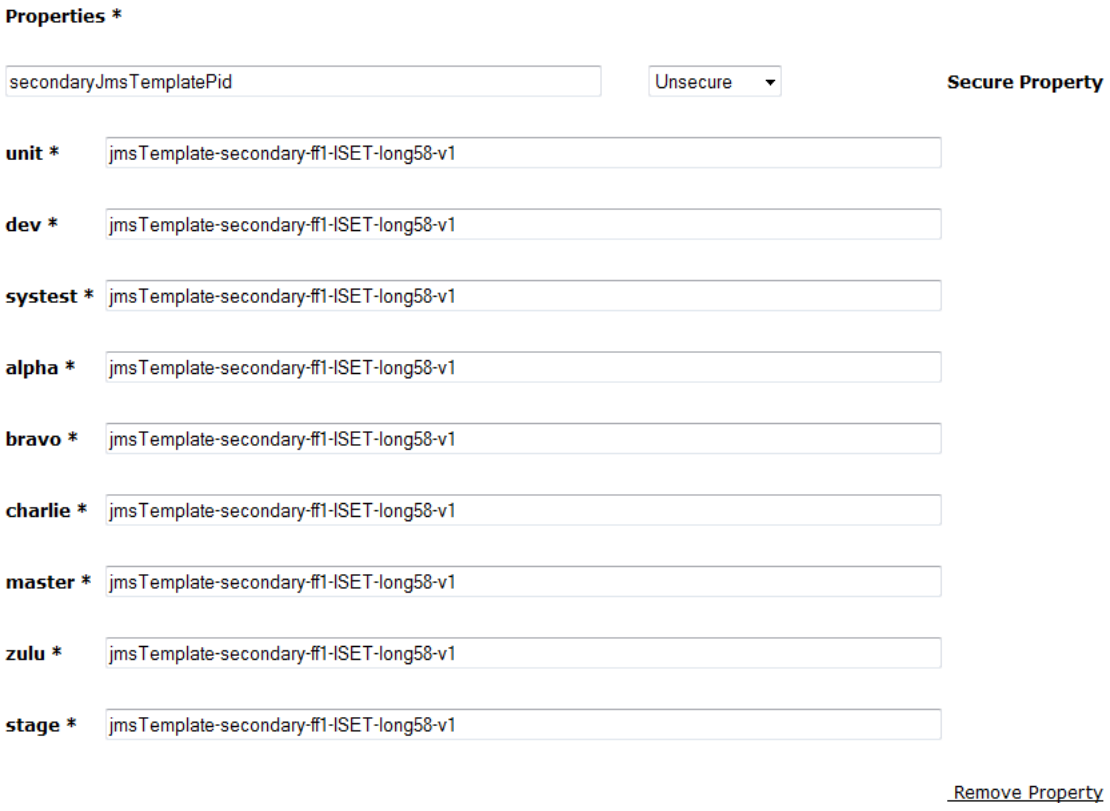
jmsSenderConfigBeanPid will have five config pid which will further have configuration properties associated with these config pid. Each config pid will be created via Create Configuration and viewable from View Configuration in upm3 config app.

i) **destinationQueuePid :-** This configuration pid have request queues configuration properties. Type this pid in view configuration search box to see underlying properties of this pid.

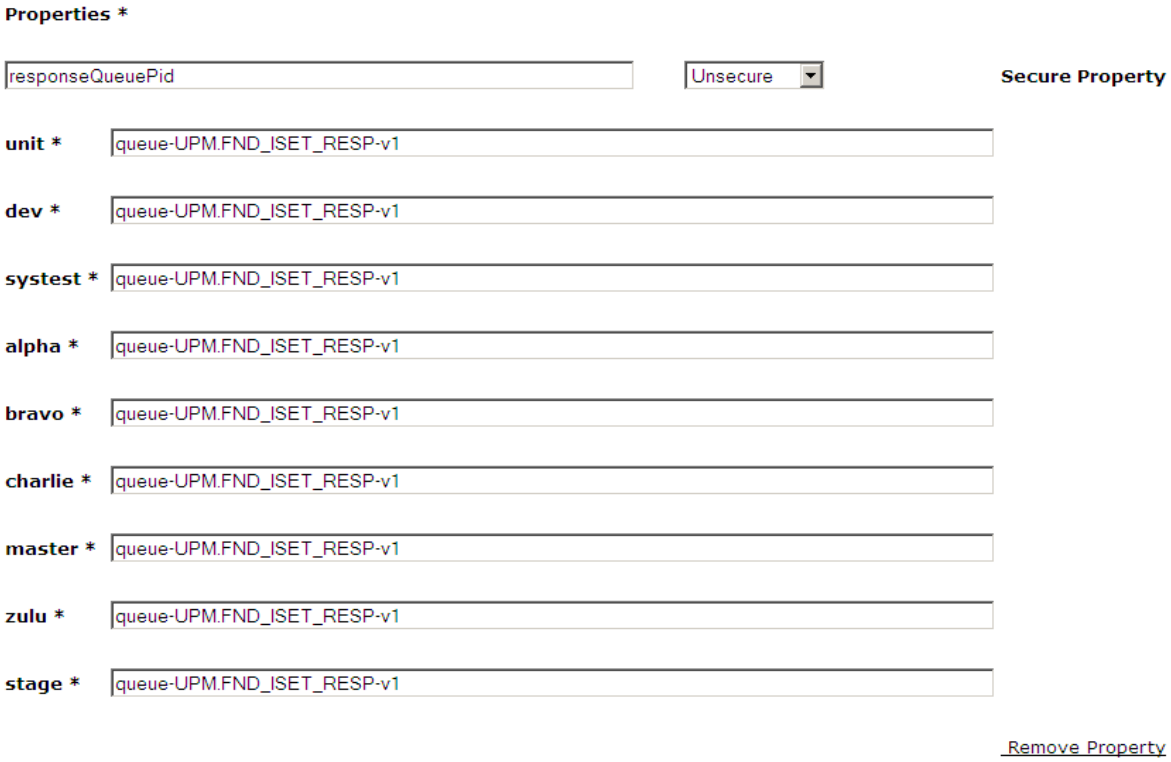
ii) **primaryJmsTemplatePid :-** This config pid will have jmsSender primary template configuration properties.



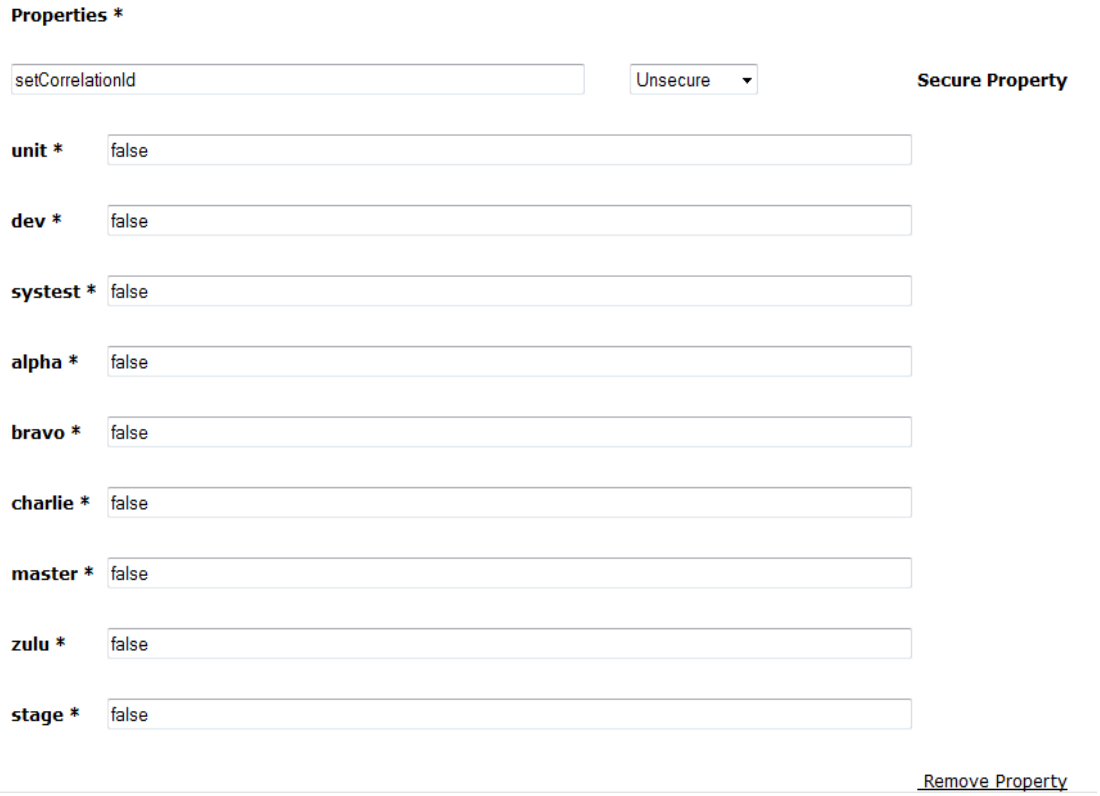
iii) secondaryJmsTemplatePid:- This config pid will have jmsSender secondary template configuration properties.



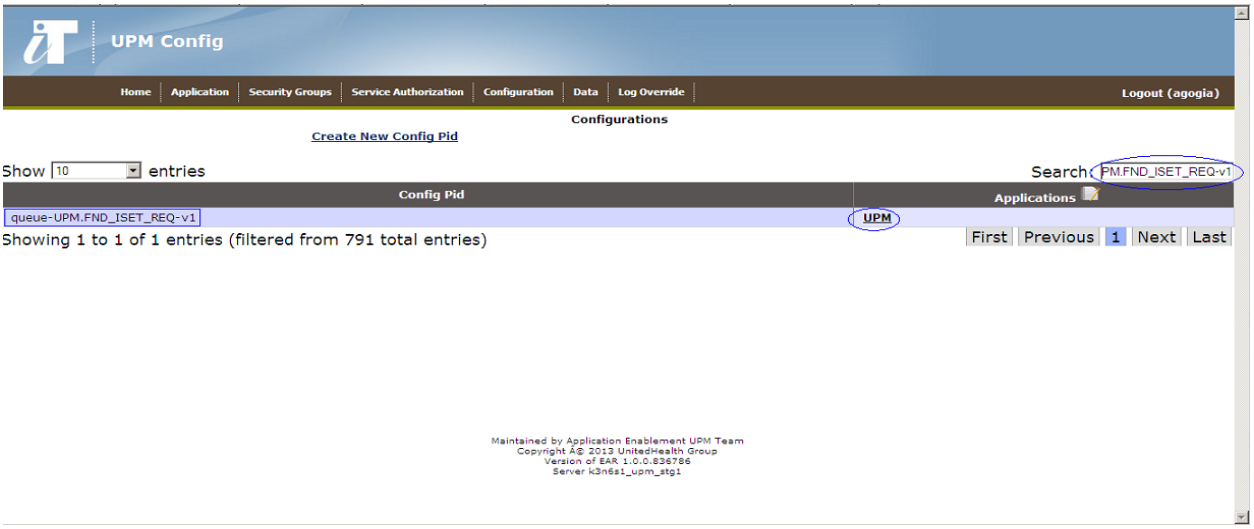
iv) **responseQueuePid** property **:-** This config pid which will have response queues configuration properties



v) **setCorrelationId** property :-

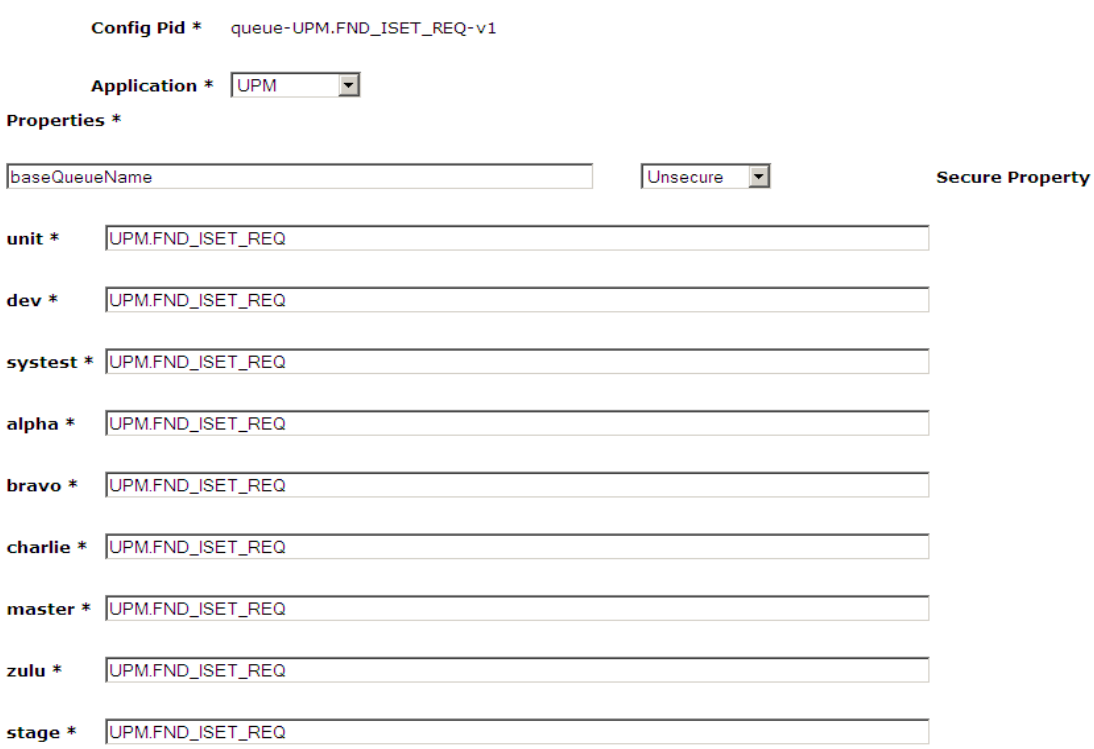


A) destinationQueuePid :- Search for request config pid, if it exists then we are good to refer else we should create.

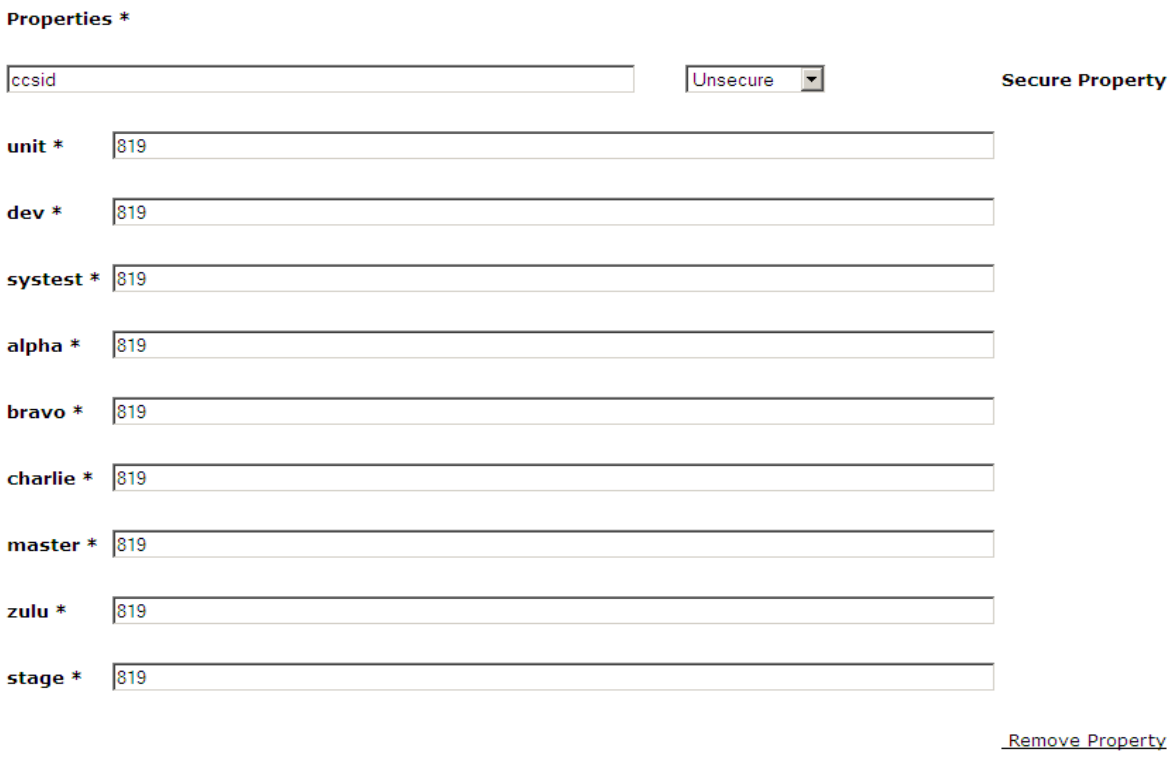


Request config pid :- (for e.g queue-UPM.FND_ISET_REQ-v1) will have five configuration properties.

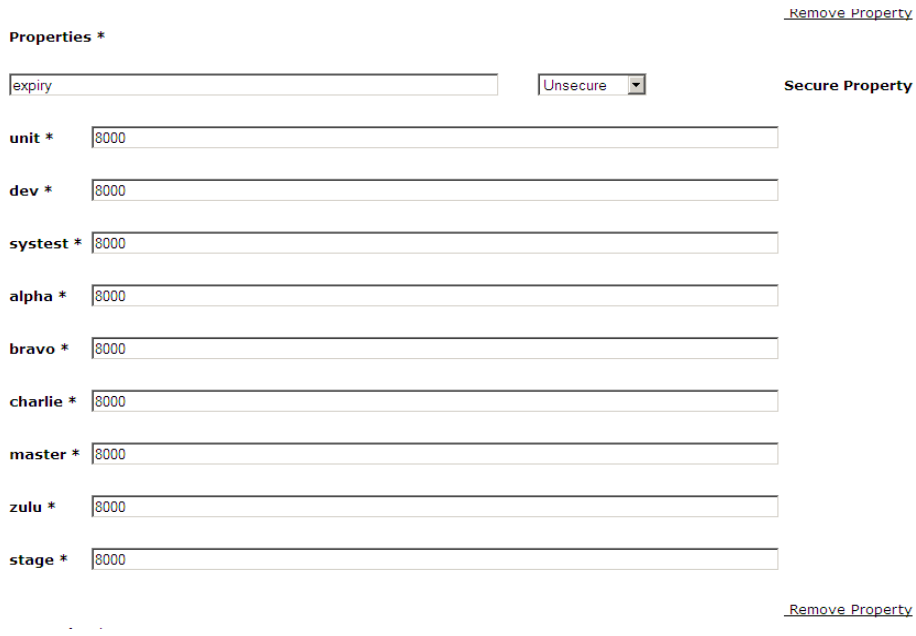
i) **baseQueueName :-** request queue name



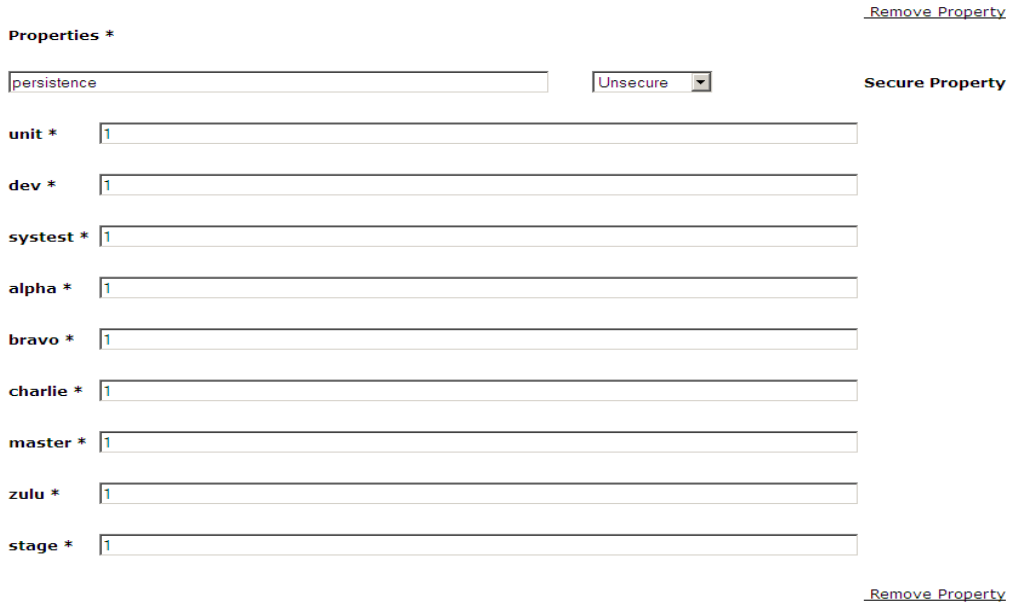
ii) **ccsid :-** default value as 819



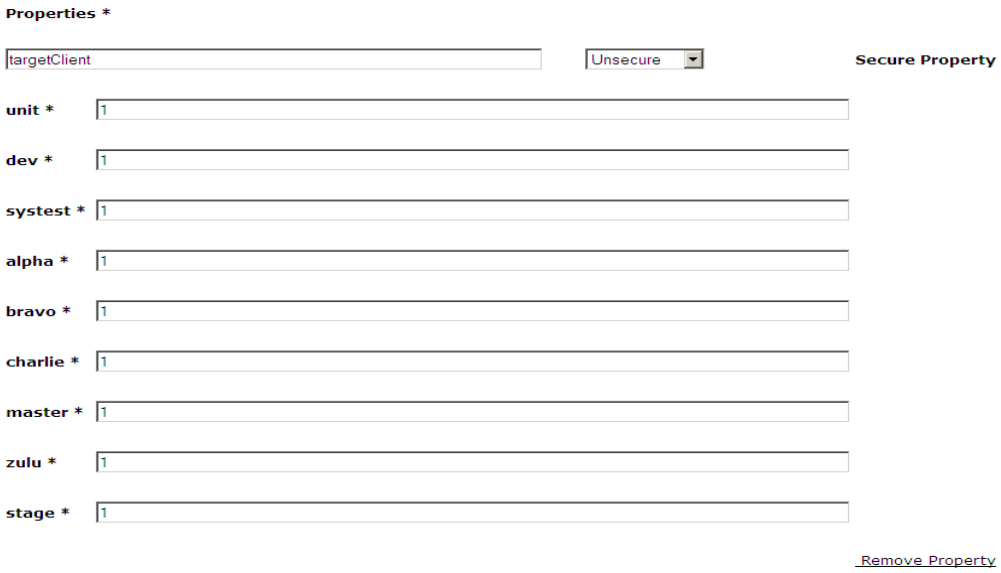
iii) expiry :- default value as 8000ms



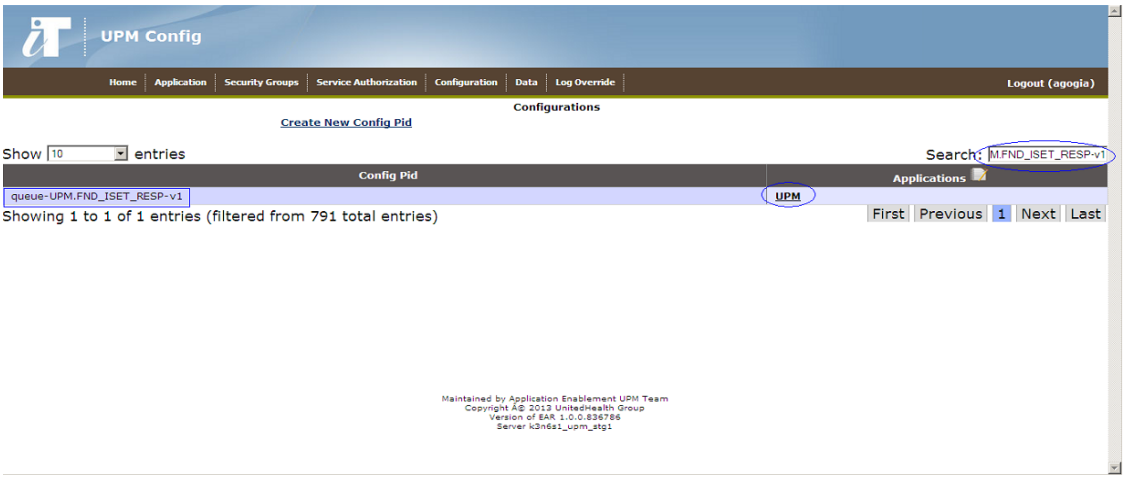
iv) persistence :- default value as 1



v) targetClient :- default value as 1

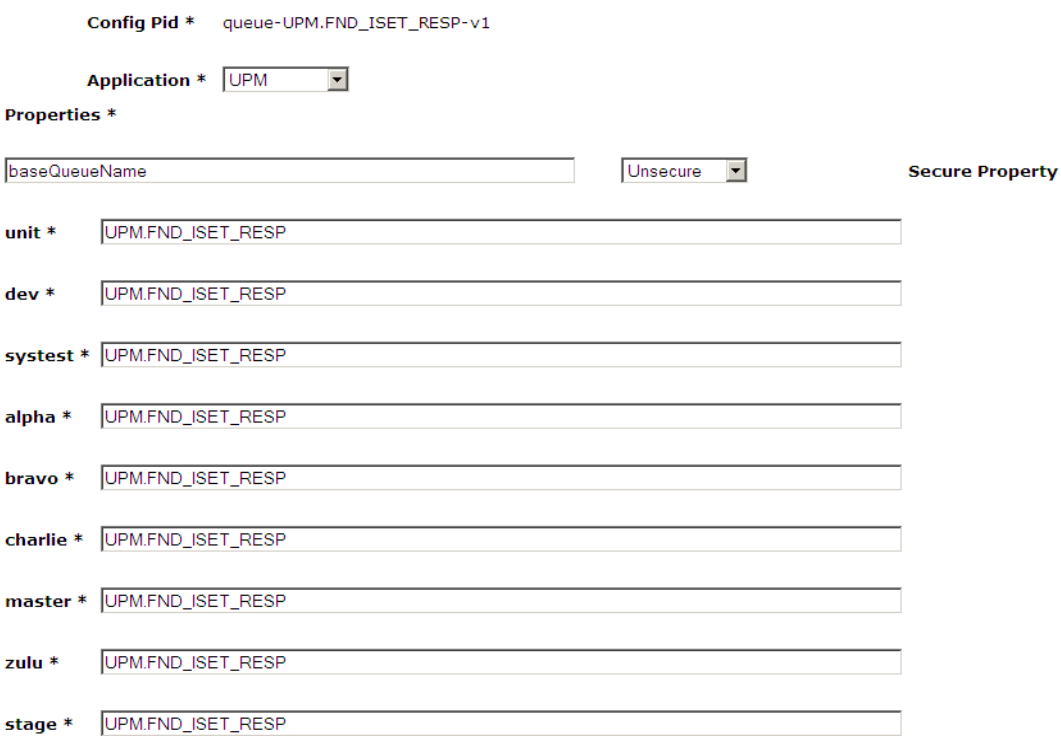


B) responseQueuePid :- Search for Response configuration pid in view configuration, else create through Create Configuration

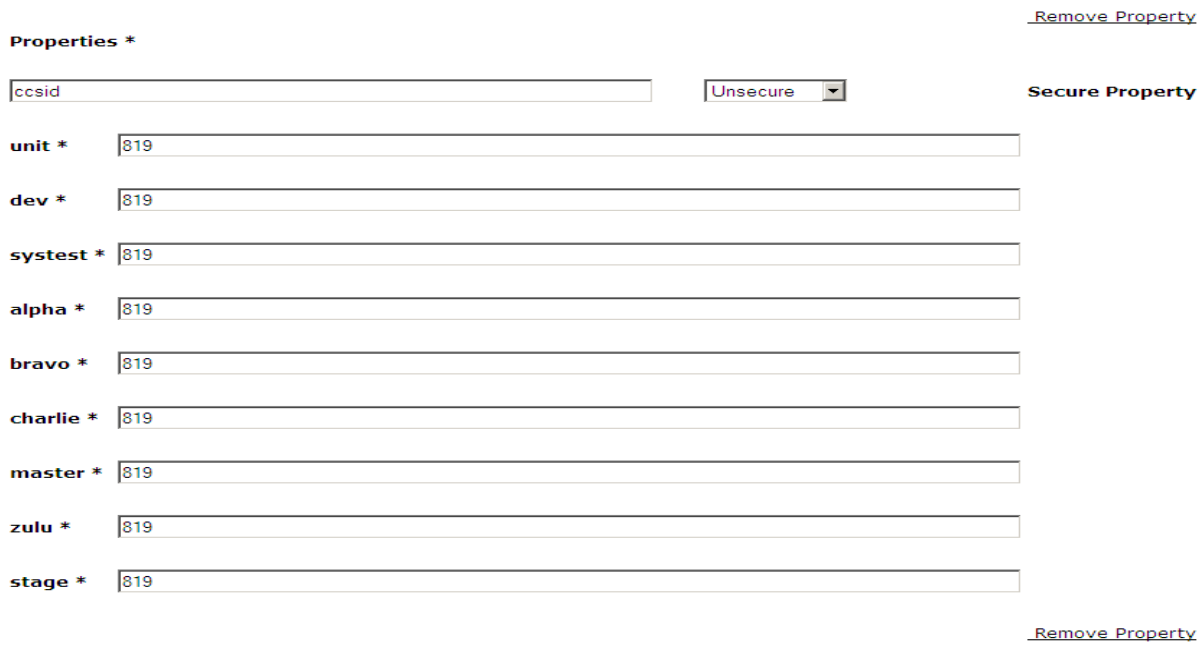


Response config Pid :- (for e.g queue-UPM.FND_ISET_RESP-v1) further have five configuration properties

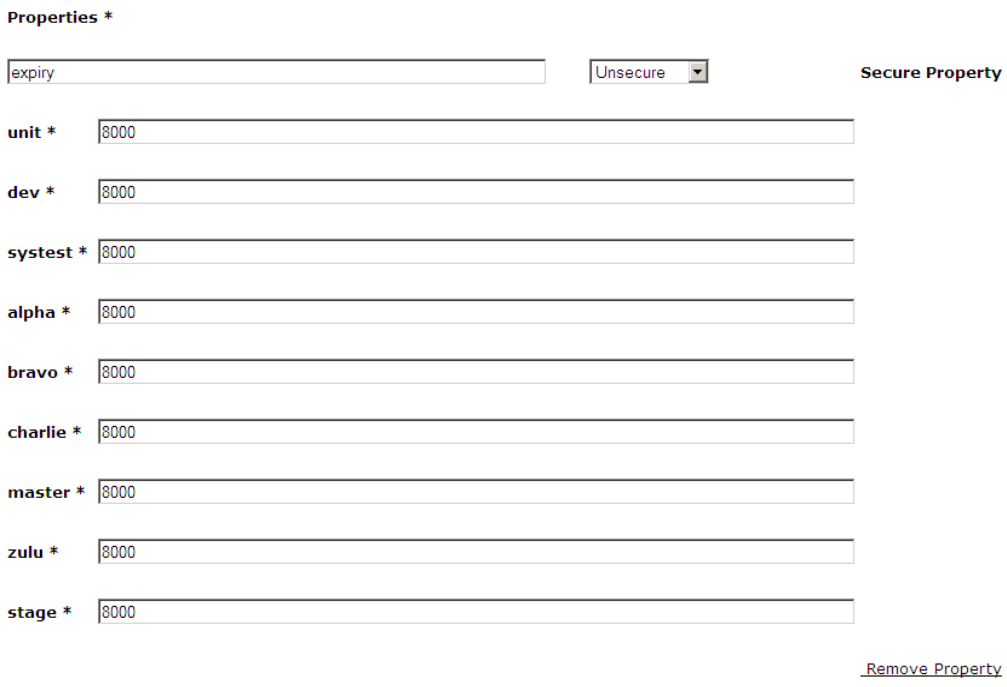
i) baseQueueName



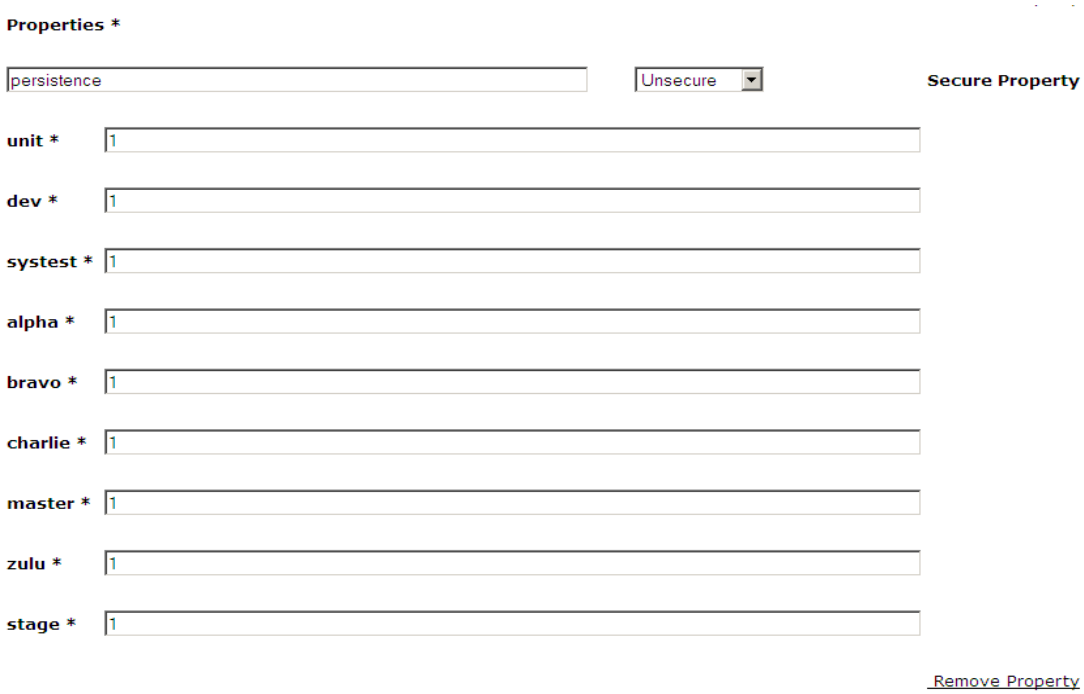
ii) ccsid :- default value as 819



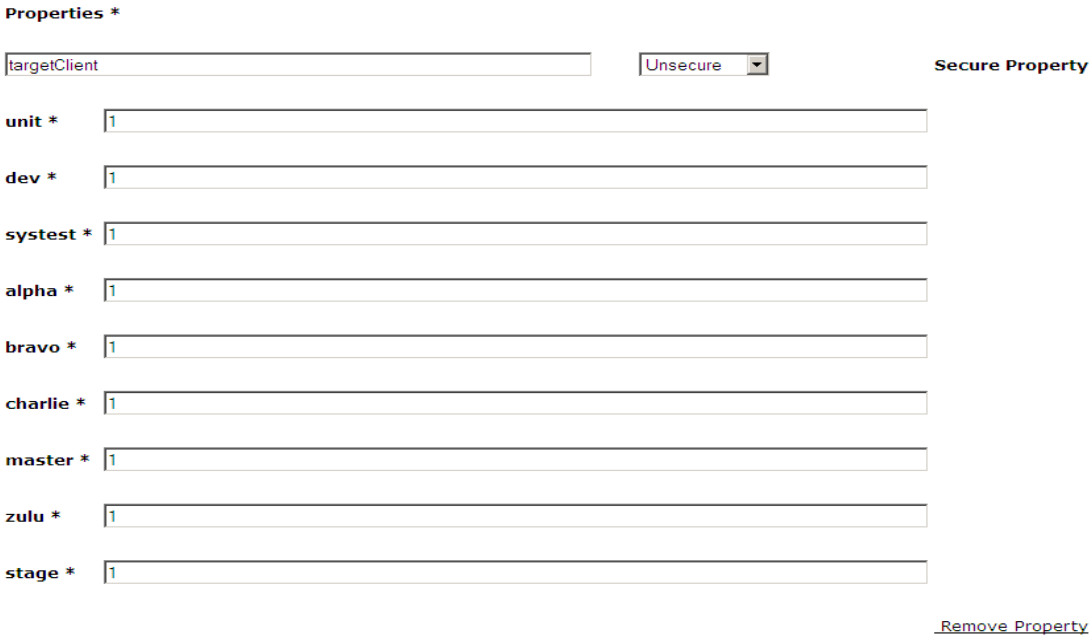
iii) expiry :- default value as 8000 ms



iv) persistence :- default value as 1



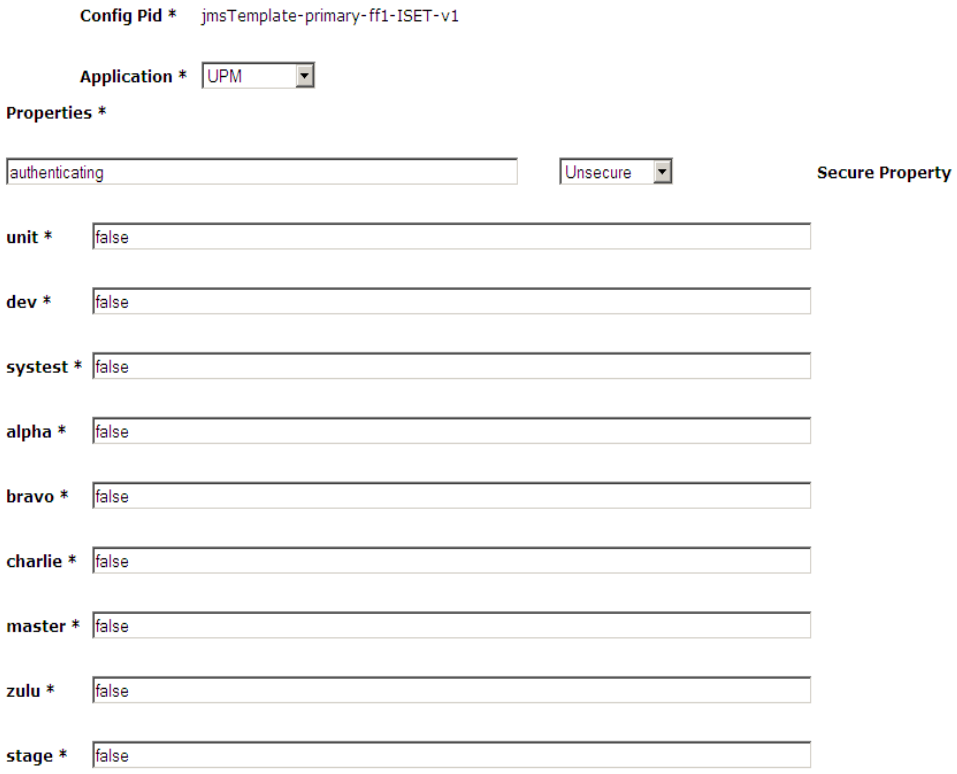
v) **targetClient :-** default value as 1



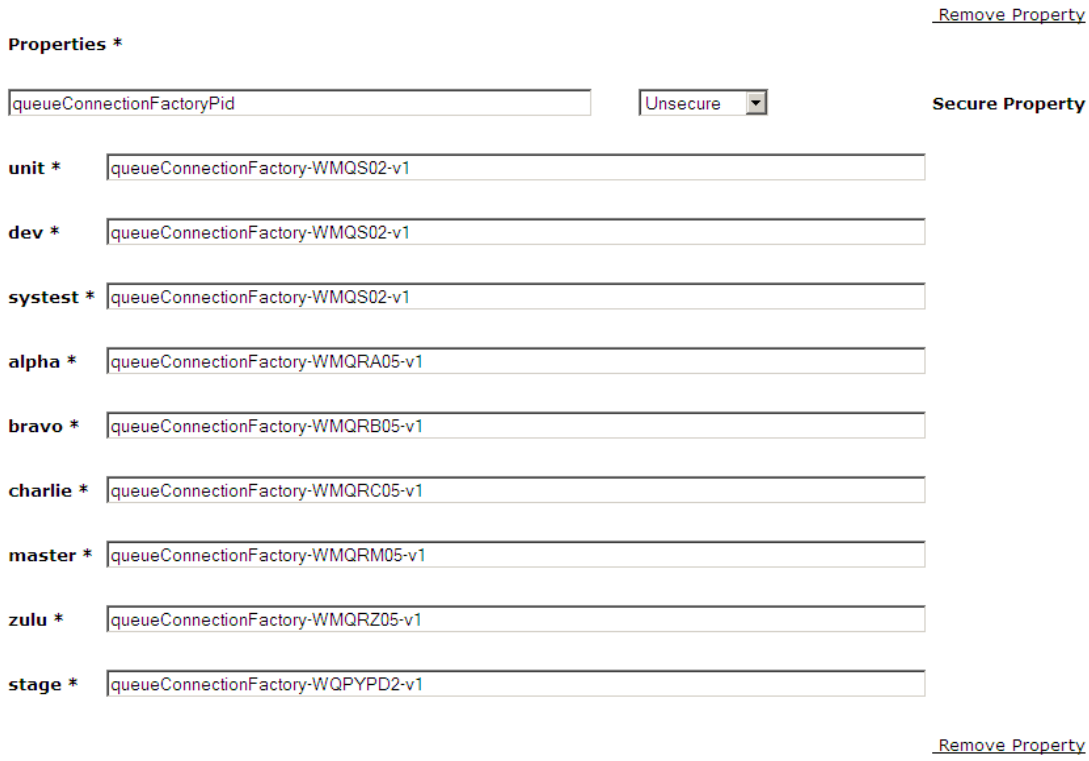
C) Search for **JmsTemplate-primary** in view configuration if its exists then verify it have all configuration properties and if it doesn't exist then create through Create Configuration in upm3 config app

jmsTemplate config pid will have three configuration pid.

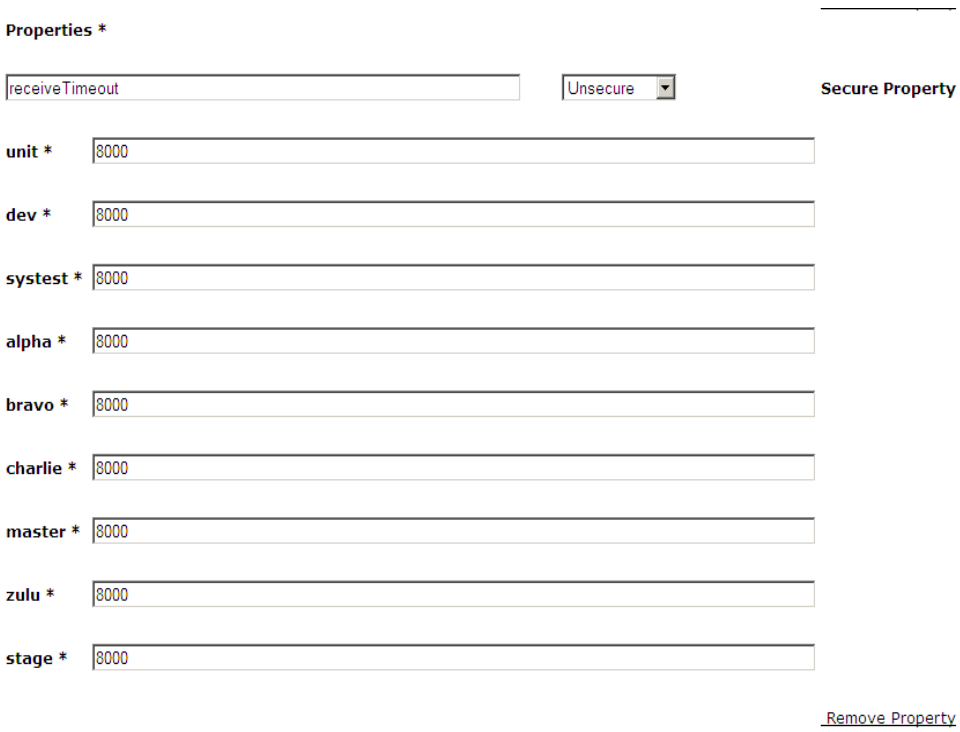
i) **authenticating :-** default value will be false.



ii) queueConnectionFactoryPid :- will have wmb queue manager info



iii) receiveTimeout :- default value as 8000 ms



Note: **For every pid for WMB service please create Primary and Secondary JMS TEMPLATES.**

Sample configuration specific for ISET ff1.

Service config pid :-

com.uhg.upm.dataservice.cdb.v6435iseunet-v2			
key	**jmsSenderConfigBeanPid**	**callingApplication**	**Application**
unit	jmsSender-ff1-ISET-v1	ISET	ISET
dev	jmsSender-ff1-ISET-v1	ISET	ISET
systest	jmsSender-ff1-ISET-v1	ISET	ISET
alpha	jmsSender-ff1-ISET-v1	ISET	ISET
bravo	jmsSender-ff1-ISET-v1	ISET	ISET
charlie	jmsSender-ff1-ISET-v1	ISET	ISET
master	jmsSender-ff1-ISET-v1	ISET	ISET
zulu	jmsSender-ff1-ISET-v1	ISET	ISET
stage	jmsSender-ff1-ISET-v1	ISET	ISET

JmsSender config pid:-

jmsSender-ff1-ISET-v1			
key	**destinationQueuePid**	**primaryJmsTemplatePid**	**responseQueuePid**
unit	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
dev	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
systest	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
alpha	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
bravo	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
charlie	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
master	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
zulu	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1
stage	queue-UPM.FND_ISET_REQ-v1	jmsTemplate-primary-ff1-ISET-v1	queue-UPM.FND_ISET_RESP-v1

Request Config Pid :-

queue-UPM.FND_ISET_REQ-v1					
key	**baseQueueName**	**ccsid**	**expiry**	**persistence**	**targetClient**
unit	UPM.FND_ISET_REQ	819	8000	1	1
unit	UPM.FND_ISET_REQ	819	8000	1	1
dev	UPM.FND_ISET_REQ	819	8000	1	1
systest	UPM.FND_ISET_REQ	819	8000	1	1
alpha	UPM.FND_ISET_REQ	819	8000	1	1
bravo	UPM.FND_ISET_REQ	819	8000	1	1
charlie	UPM.FND_ISET_REQ	819	8000	1	1

master	UPM.FND_ISET_REQ	819	8000	1	1
zulu	UPM.FND_ISET_REQ	819	8000	1	1
stage	UPM.FND_ISET_REQ	819	8000	1	1

jmsTemplate primary and secondary config pid

jmsTemplate-primary-ff1-ISET-v1			
key	**authenticating**	**queueConnectionFactoryPid**	**receiveTimeout**
unit	false	queueConnectionFactory-WMQS02-v1	8000
unit	false	queueConnectionFactory-WMQS02-v1	8000
dev	false	queueConnectionFactory-WMQS02-v1	8000
systest	false	queueConnectionFactory-WMQS02-v1	8000
alpha	false	queueConnectionFactory-WMQRA05-v1	8000
bravo	false	queueConnectionFactory-WMQRB05-v1	8000
charlie	false	queueConnectionFactory-WMQRC05-v1	8000
master	false	queueConnectionFactory-WMQRM05-v1	8000
zulu	false	queueConnectionFactory-WMQRZ05-v1	8000
stage	false	queueConnectionFactory-WQPYPD2-v1	8000

jmsTemplate-secondary-ff1-ISET-v1			
key	**authenticating**	**queueConnectionFactoryPid**	**receiveTimeout**
unit	false	queueConnectionFactory-WMQS12-v1	8000
unit	false	queueConnectionFactory-WMQS12-v1	8000
dev	false	queueConnectionFactory-WMQS12-v1	8000
systest	false	queueConnectionFactory-WMQS12-v1	8000
alpha	false	queueConnectionFactory-WMQRA12-v1	8000
bravo	false	queueConnectionFactory-WMQRB12-v1	8000
charlie	false	queueConnectionFactory-WMQRC12-v1	8000
master	false	queueConnectionFactory-WMQRM12-v1	8000
zulu	false	queueConnectionFactory-WMQRZ12-v1	8000
stage	false	queueConnectionFactory-WQPYPD06-v1	8000

Response config pid

queue-UPM.FND_ISET_RESP-v1					
key	**baseQueueName**	**ccsid**	**expiry**	**persistence**	**targetClient**
unit	UPM.FND_ISET_RESP	819	8000	1	1
unit	UPM.FND_ISET_RESP	819	8000	1	1
dev	UPM.FND_ISET_RESP	819	8000	1	1
systest	UPM.FND_ISET_RESP	819	8000	1	1

alpha	UPM.FND_ISET_RESP	819	8000	1	1
bravo	UPM.FND_ISET_RESP	819	8000	1	1
charlie	UPM.FND_ISET_RESP	819	8000	1	1
master	UPM.FND_ISET_RESP	819	8000	1	1
zulu	UPM.FND_ISET_RESP	819	8000	1	1
stage	UPM.FND_ISET_RESP	819	8000	1	1

II. Configuration for web service based backend

We need to create two configuration properties i) endpoint and ii) timeout.

Properties *

timeout | Unsecure | **Secure Property**

unit * 8000

dev * 8000

systest * 8000

alpha * 8000

bravo * 8000

charlie * 8000

master * 8000

zulu * 8000

stage * 8000

Remove Property