## 1.0 Introduction

This document gives a high level overview of proposed SVC application architecture. This architecture consists of the following three interfaces

* Oracle to MongoDb interface – see section 2.0
* MongoDb to SVC Web Application interface – see section 3.0
* SVC Web Application to MongoDb interface – see section 4.0

## 2.0 Oracle to MongoDb Interface

An API has been developed to load data from any Oracle table (or view) into any MongoDb database or collection. The API is a Linux command line interface consisting of the following commands:

* test\_oracle\_mongo – this acts as a basic health check on ETL code. It verifies successful set of ETL infrastructure
* verify\_oracle\_mongo – this verifies that ETL process can successfully process the data from a specific Oracle table. It run the full ETL code for a given Oracle source table except insert or update of Mongo. This acts as a health check on the Oracle data source
* insert\_oracle\_mongo – this does a full load from an Oracle data source into Mongo. It assumes and emtpy target Mongo collection so that all data is loaded into Mongo in insert mode.
* update\_oracle\_mongo – this command is designed for the daily incremental load from an Oracle data source to Mongo. If the Oracle row does not exist in Mongo it is inserted – otherwise the Mongo row is updated.

## 3.0 API from MongoDb to SVC Web Application

The API from MongoDB to SVP Web Application consists of a single Mongo document per country. This document contains the list of values and counts for all the segmentation fields. In the initial release information is available for the following segmentation fields:

* LAST\_REGION
* MOST\_RECENT\_PERMISSION
* MOST\_RECENT\_GENDER
* MOST\_RECENT\_BRAND

This summary level information is stored in the Mongo database SVC\_SUMMARY collection. The summary data is refreshed on the VM by means of the following linux command:

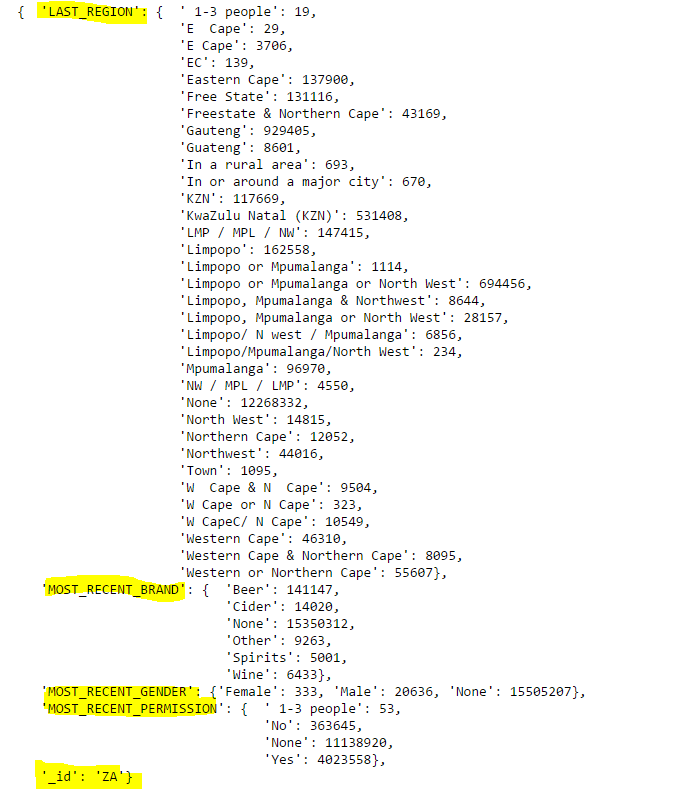
* For South Africa the command is: summarize\_country\_collection ZA
* For Indonesia the command is: summarize\_country\_collection ID

The information is available from the following web services APIs:

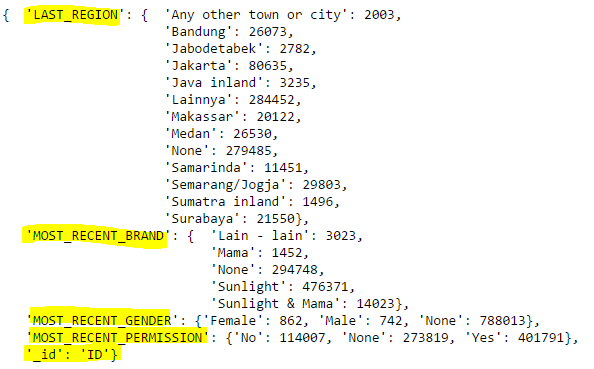
* For South Africa the url is: http://10.177.177.60/svc/v1/za/summary/
* For Indonesia the url is: http://10.177.177.60/svc/v1/id/summary/

Please note that the information available from the web services is intended for consumption by a web application – for example a java application like New Konnect. As a result the data format is not pretty printed, however is ideal for processing by modern software programs such as Jave or Python. Figure 1 and 2 show the data produced by the web services but passed through pretty print API in order to make the data more easily read by human beings.

***Figure 1: Country Summary for South Africa. This data is from Mongo and available from web services API. This is how this data looks when it is formated for human inspection.***



***Figure 2: Country Summary for Indonesia. This data is from Mongo and available from web services API. This is how this data looks when it is formated for human inspection.***



## 4.0 SCV Web Application to MongoDb Interface

The user story is that we define the segment as follows. Find all MSISDN for South Africa where

* Region is either "E. Cape" or "LMP / MPL / NW"
* Permissioned is “Yes”
* Gender is “Male”
* Brand is “Beer”

A Mongo composite index was created on the following 4 fields: "LAST\_REGION", "MOST\_RECENT\_PERMISSION", "MOST\_RECENT\_GENDER", "MOST\_RECENT\_BRAND”.

Here is the Python request to find above information:

db.DW\_SVC\_ZA.find ( {

"LAST\_REGION": {"$in": ["E. Cape", "LMP / MPL / NW"]} ,

"MOST\_RECENT\_PERMISSION":"Yes",

"MOST\_RECENT\_GENDER":"Male",

"MOST\_RECENT\_BRAND": {"$in": ["Beer"]

} }, {"\_id": 1} )

The Mongo shell responds immediately with the 5 MSISDNs that meet the above criteria. The response is immediate and is as follows:

{ "\_id" : "27713902768" }

{ "\_id" : "27715228992" }

{ "\_id" : "27769637896" }

{ "\_id" : "27786946034" }

{ "\_id" : "27790550013" }

Commentry:

When SVC application issues a request and if Mongo indexing is correct the response from Mongo should be immediate. However there is a risk that the SVC application will issue a request that causes a full Mongo table scan. In order to mitigate the Mongo full table scan risk, it would be best to implement the SVC Web application requests as a web service. Such a web service would log all SVC application requests. It could queue all request – and reject requests that are known to cause full table scans. It would act as a fire wall between the web application and Mongo. Python is an ideal language to create such a web service.