**TDM (Test Data Management)**

TDM portal is for serving the Test Data. It will create the Data and to give the ATOM framework, which means it will not actually create the data, it will fetch available data from the TDM data base on basis of user selections like Env, TechType, datatype etc.

* And basically there are 3 types of users in TDM portals.

1. Admin
2. User
3. Service

* Admin can have the access to see data statistics
* User have the basic access like fetching the data out from the portal.
* Service – TBD (not sure)

Stats Api:

This api will fetch the information about the Vacant available data.

***Checkout Functionality***:

Checkout functionality is basically checkingout the data from TDM database which are ready to use for testing.

If the product is Mobile then it will checkout the data from Mobile and Mobile\_Msisdn tables and join them and finally return.

As soon as it fetching the data checkout id will be generated and update the checkout\_id into both Mobile and Mobile\_Msisdn tables. So that this will not be available for next time check out. Which means if there is checkout\_id then we can consider the mobile data has been already used.

There are two different data can be checkout from the TDM.

* Vacant Available Data
* Active Data

***Vacant Available Data***:

* To fetch the vacant data from the TDM portal we have to give the below details as input
  + Env – from which env to fetch the data – INT1 / INT2
  + Product: User has to choose for which product data should checkout – Mobile, NBN, ADSL, Cable
  + Technology: This technology will display based on selected in the product.

Let’s says, if the product is selected Mobile, then the Technology will displays the Hybrid and Postpaid.

* + Quantity: number of available records should be checked out.

***Technical Flow***:

Once give all the parameters and click on Checkout, it will send the request to /checkout api in DownloadController.

* DownloadController – Checkout api will receive vacantCheckoutRequest object which contains the input data and Authentication object.
* It will call the checkout method of the DownloadService.
* Check the valid user, otherwise return the usernotfound exception.
* Now call the checkout method from the specific service on basis of Product which we have selected. If the product is ‘Mobile’ then call the checkout method of the VacantMobileDataService, if the product is other than Mobile for example NBN, ADSL, Cable, Transition then call to checkout method from VacantAddressDataService.

**Product: Mobile**

If the product is Mobile then call to checkout method from the VacantMobileDataSerivce.

**Checkout Mobile data**: Fetch the mobile data from the MOBILE table by passing input parameter and pass the team id is 111 (hardcoded value) as a team id.

The team id as of now passing the hard coded value 111, even though if the user from some other teams like 999, 222 still fetching the vacant data from 111 only for now.

**Checkingout Msisdn data:**

For checking out the msisdn data there two tech types, PostPaid and PortIn.

If the TechType is Postpaid, Fetch the available data by passing the input parameters and also team is 111 (hardcoded). Once fetch the msisdn data, we have to check whether that data contains the checkoutid in the database, if there then we need to clean/refresh the checkout id. Currently this cleanup functionality is available only when techtype is Postpaid. Not available for other techtypes.

If the TechType is PortIn, now check the foreign range from the SNRM, if the range is available in snrm then take that msisdn data.

Once get the postpaid data and msisdn data join both of them.

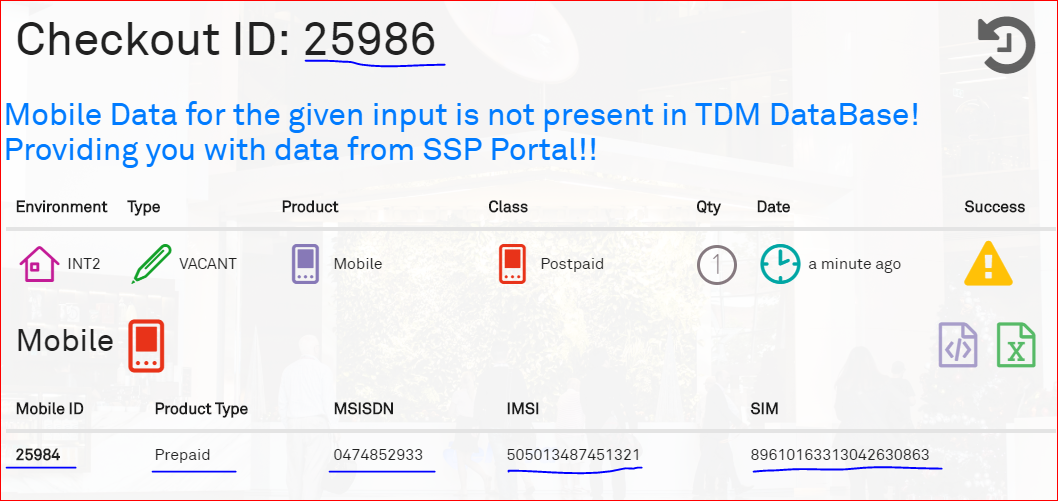
If the postpaid and msisdn, are empty (no data in TDM database) then go to the SSP (self service ports) and fetch the data. Here SSP is the external service to call and get the response.

SSP URL for the reference: http://w0e1d37ae1db53.wg.dir.telstra.com:8080/SSP/WS/JSON

Mobile checkout for the specific resources which are msisdn, sim/imsi or all, it will assign accordingly.

And finally it will return the checkout log to client.

**The final o/p for checkout the mobile data as below,**



**Product: NBN, ADSL, Cable, Transition**

Check if the datatype value is any of the value from NBN, ADSL, Cable and Transition then call the checkout from the VacantAddressDataService.

For the NBN checkout we have to give below input values

* techtype – is an integer value, which will be technically the classId from the CONFIG\_NBN\_CLASS\_NAME\_MAPPING
* datatype – this is the product type like NBN, Cable etc.
* qtyType – No’ of records to be checked out
* fnn
* tagIds
* envType – Env values like INT1 or INT2 etc
* className – It’s type of the technology like FTTP, HFC, FW, FTTN, FTTB, FTTC.

Once send the request to DownloadController with input values, the controller will hand over the request to DownloadService checkout. This service will find out the actual AddressVacantDataService with datatype and call the checkout method of AddressVacantDataService.

Now check if the tagIds is true then call the checkout functionality for the specific product.

If suppose the product is NBN, then call the checkOutNbnData method. Now get the available data from VACANT\_ADDRESS and CONFIG\_NBN\_CLASS\_NAME\_MAPPING by passing serviceabilityClass, className, nbnCapability and goodToUse should be 1.

While fetching the data, check className given as input or not, if yes then pass the className to query and get the data otherwise fetch the data when goodtoUse is 1, nbnCapability is true, env and serviceAbilityClass ids.

If the product is Cable, then call the checkOutCableData. Now get the data by from VACANT\_ADDRESS and CONFIG\_NBN\_CLASS\_NAME\_MAPPING by passing the goodtoUse is 1, AdslCapability is true, env and CramerCapability is true.

If the Product is Adsl, then call the checkoutAdslData, now get the data from same VACANT\_ADDRESS and CONFIG\_NBN\_CLASS\_NAME\_MAPPING tables, by passing the values goodtoUse is 1, AdslCapability is true, env.

If the Product is Transition, then call the checkoutTransitionData, now get the same VACANT\_ADDRESS and CONFIG\_NBN\_CLASS\_NAME\_MAPPING tables, by passing goodtoUse is 1, nbnCapability is True, AdslCapability is True and serviceAbilityClass ids.

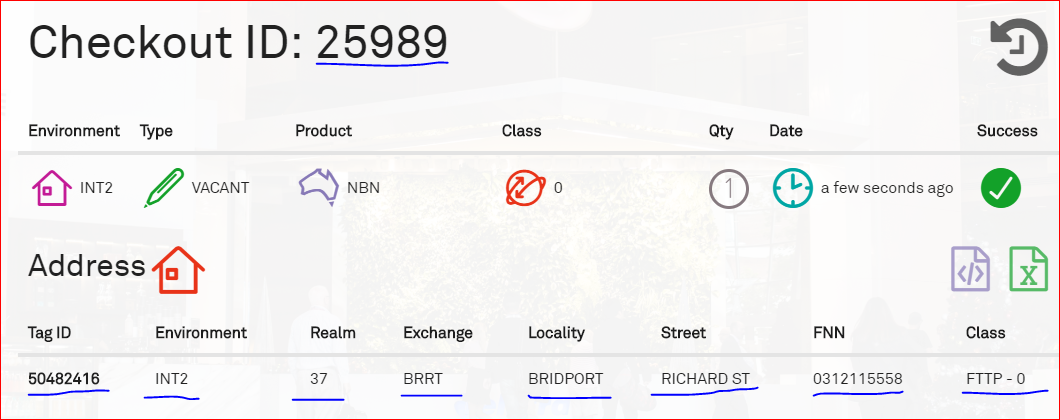
Once get the Address list back from the above steps, then pass this list to checkout log.

Check the datatype is NBN or Transition and fnn is true, then call then checkOutFnnsWithEnv in VancanFnnDataService, get the available FNNs . Once get the FNN items check the addressItems are more than or equal to fnnItems. If yes, then assign the fnns into nbn address data.

Check address is success and fnn is success then set checkout log success is true.

And finally save the checkout log in to the CHECKOUT\_LOG table after setting all the data which needs to be returned and then return back to client.

Below is the screenshot of the data after checkout,



Here tag id fetch from VACANT\_ADDRESS – TAG\_ID

Environment is which environment was expected to checkout the data

Realm and Exchange is for State and area.

Locality and Street, these are address related values of the customer

FNN – is the mobile number of the NBN

Class – it is the type of the broad band FTTP (Fiber to the Premises).

***Active Data****:*

Active Data module will be connected to the downstream applications to make the vacant data into active data. The Data will be comes the active data when it’s linked to any customer, this is same either for Mobile data or for NBN data.

There are two different data types, Order and Customer.

**Data Type - Customer**:

For the customer data types, we can create the different type of customer types from the various external downstream applications.

* Residential, Sole Trader, Company, Organization customer’s creation done by the Seibel downstram application.
* TB unmanaged customer creation done by MICA downstream application.

Under the Residential, Sole Trader, Company, Organization customer types there are two subtype customer types which are Realtime and Existing.

Realtime subtype means, create the customer done by the Seibel application, and whereas existing customer subtype means get the existing created customer from the TDM api and map to the vacant data for activate.

Under the TB unmanaged customer have one subtype which is Realtime where we can connect to the MICA legacy IBM system to create the customer.

***Technical Flow:***

* When select the below options
  + Environment: **INT2**
  + DataType: **Customer**
  + CustomerType: Residential | Sole Trader | Company | Organization
  + Customer Subtype: Realtime

At this time customer creation will be done by the Seibel downstream application.

Steps for the technical flow of Seibel application:

* Once select all the above options and click on checkout, then ActiveDataController – checkout service will get call.
* This will hand over the request to checkout method from ActivateDownloadImpl by passing the input / selected data in the form of ActiveCheckoutRequest object.
* Now pass the optionType is fetch for customer data of the specific option. Here optionType nothing but it is unique number which we have got from the filter service by passing the Customer DataType (Residential) and DataType (Customer) and SubType (Realtime) from the table called CONFIG\_ACTIVE\_PLAN\_MAPPING.
* Check the user authentication, by checking the user availability.
* Call the checkout method of the ActiveCustomerService by passing the checkoutRequest object (which contains input data) and ActiveRequestmapping (config active plan mapping).
* Check the max quantity of that given request type, if the request type is Realtime when the max quantity is 1, and if the request type is Existing then the max quantity is 20. So if the requested quantity is more than max quantity then throw the fatal exception by saying supports only max quantity.
* Since the selected subtype is Realtime, then it will call the checkOutCustomerRealtime functionality where we have the Seibel code to create the customer.
* Call the createCustomer to create the new ActiveCustomer by passing the environment type and config mapping object.
* Now Check the classname(CustomerType) is TB Managed then call the micaCustomer flow otherwise call Siebel customer creation flow. So here we can see the Siebel customer creation flow.
* Create the some user details like firstname, lastname, email, dob etc. and pass these details to the Seibel module service method – createCustomer.

Seibel Flow for Customer Creation:

To send the request to the Siebel to create the customer, need to pass the data in the form of xml format. So here there are some xml predefined templates for each user types. And the path of this physical xml template file path would available in the ActiveRequestmapping (CONFIG\_ACTIVE\_PLAN\_MAPPING) TEMPLATEFILE column.

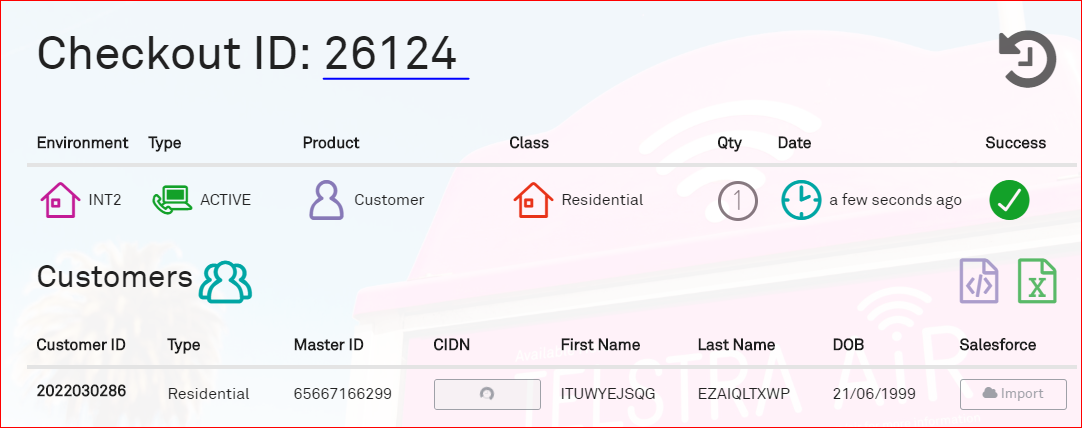
Get the xml template file and parse that xml file with our customer created details, by calling the parseXMLRequestToString in the SeibelXmlService. Then it will returns the xml format data in the form of string.

Now pass this string to the Siebel Message queue with authentication values (username and password of the Siebel application). Then Siebel will execute the message from the queue then it will create the customer.

Wait for some time by thread sleep, to make sure the Siebel execute the message queue value and created the customer.

Now fetch the created customer from the Siebel by calling the findCustomer from SeibelDataService by passing the customer details as input (firstname, lastname, email etc).

Check if there is customer fetched, if yes then return the customer to the client otherwise send the appropriate message saying issue in xml template.



* When select the below options
  + Environment: **INT2**
  + DataType: **Customer**
  + CustomerType: Residential | Sole Trader | Company | Organization
  + Customer Subtype: Existing

In this case the customer will not create, it will get the existing customer from the TDB data base and return it.

*TO BE CHECK (analysis) for more details on this case (topic)*

Now connect to the MICA application to create the customer for the customer type TB unmanaged.

Steps for the technical flow of MICA application:

* When select the below options
  + Environment: **INT2**
  + DataType: **Customer**
  + CustomerType: TB unmanaged
  + Customer Subtype: Realtime

For the MICA all the steps are equal till checkoutCustomerRealTime. Inside the createCustomer, checking the className (CustomerType) is TB unmanaged, if yes then call the micaCustomer where we are actually creating the customer via MICA downstream application.

Call the MicaDataService – createCustomer by passing the env, and there create the customer details and pass them to createCustomerAndBillingAccount where actually the customer created. And here to make the call the MICA IBM application. This service is available in the TDM framework (need to get more details about tdm framework to elaborate more on this topic).

**DataType - Product:**

Once the available vacant checkout data is active by assigning it to customer then the sim should activate with some value added services. Which means plans which are providing by the Telstra should be updated to the mobile, then user can use it.

Here there are few steps will follow to checkout the product related active data.

* + Create the Customer
  + Create the Billing Account number and assign it to the Customer
  + Call the ContactBundle to get the contactId to map with customer and order.
  + Create the order

Under the order product the Mobile order type will be there, once select the mobile type then it will show two options, create the order via Siebel downstram application or MICA application.

**Seibel**:

If the order type is selected the PostPaid – SIEBEL, then it will show the order subtypes, which are actually mobiles plans for the applicable to the active mobile data.

* When select the below options
  + Environment: INT2
  + DataType: Order
  + ProductType: Mobile
  + Order Type: **PostPaid – SEIBEL**
  + Order Subtype: anyone (Mobile BYO Plan $49)

Once select above details and checkout, it will go the checkout service from ActiveDataController. And it will handover it to checkout method of ActiveDownloadServiceImpl, and here also get active mapping data (CONFIG\_ACTIVE\_PLAN\_MAPPING) by passing the option type, and after that check the authentication of the give user (validation of the checking known user).

* Call the checkout functionality of the ActiveOrderService, from here it will go to the createOrder since we are creating the order in this flow.

Here it will check the request came for NBN, Cable or Mobile, but as of now in TDM we have only Mobile to creating the order. So it will call to the createMobileOrder for creating the Order.

* Get the mobile vacant data by connecting to the vacant download service and pass this vacant data to Siebel to create the customer.

After get the vacant data pass the vacant data to createMobileOrderInSeibel functionality, here get the available existing customer by passing the hard coded values Env: ‘INT2’, CustomerType: ‘Residential’ ( static values because we can just need to get some customer to create the order and checkout the order details)

* Pass the customer Id to createbillingAccount of the Seibel Module service, so that it will create the billing account for the customer, and return the billing account.
* Get the contact bundle to fetch the contact id for passing it to the order where this will use for link between customer and order.
* Update the SNRM database for the fetched available vacant data. It will update for the specific msisdn status with DISPATH HOLD in RM1RESOURCE table.

Get the xml template format from the config active data mapping table and parse the xml template with active customer and mobile data (msisdn, SIM, imsi) and also with billing account details. Once parse that xml template and pass this to the processXMLRequest from Seibel service.

This is parse the string values in to xml file and it will return the xml file, now send this xml file to insertXMl. Here it will create the sftp channel and create the order and send it back as a response.

Now log if any errors occurred and add Siebel created order id ActiveCustomer. Set this active customer to the checkout log and finally save this checkout logs and return to client.

**MICA:**

* When select the below options
  + Environment: INT2
  + DataType: Order
  + ProductType: Mobile
  + Order Type: **PostPaid – MICA**
  + Order Subtype: anyone (Mobile BYO Plan $49)

In the ActiveOrderServiceImpl it will check the order type is MICA or Seibel, if it MICA then call to createMobileOrderInMica.

Fetch the ActiveCustomer from the MICA by creating the customer, also here create the Billing Account as well and assign it to customer.

And now call the createMicaServicePlan in the MicaDataService, here it will create the URL by passing all the information required to create the order in the URL query parameter. And send this URL to IBM mica service by using the jsc (java secure channel) and return flag as per the status from the mica service.

If the service status is true (which means created the order success) then pass the order details to the Active customer and set it to checkout log.

Now save the checkout log and return it to the client.