Gherkin Scripting Guideline: Octane

Thursday, October 19, 2023 3:22 PM

Index

As of – 09-07-25

Function
Create CUR
Airtime Advance from Channel (Pheta to provide)
Voice, Data and Airtime Advance from Fusion
FreeChange
Topup "airtime" allocation verification
Add-to-Bill function
Adding and Using Vodabucks

LTE Prepaid

Introduction

As NESD-QA Test Automation team, we have identified a gap whereby there is some misalignment between Test Analysts and Test Automators when it comes to capturing on Gherkin scripts and automation thereof. The is a particular way that Test Automators expect Test Analysts would capture Gherkin scripts and this document is meant to detail exactly that.

Definitions, Abbreviations and Acronyms

Term	Definition
UNITS	Unit code for SMS, used during SMS bundle purchase on Fusion
MINUTES	Unit code for Voice, used during Voice bundle purchase on Fusion
MB,GB	Unit code for Data, used during Data bundle purchase on Fusion

Create MSISDN

For every test we would normally delete and recreate an MSISDN profile on CCS.

Below two tests have now been combined to one function

Given New subscriber <Subscriber_type> profile is <Offering_Code>

- Identifies an MSISDN to be used during test execution (<Subscriber_type> = MSISDN to be used during execution).
- Delete and the recreate the MSISDN profile on CCS as identified by <Offering_Code> parameter. E.g. <Offering_Code> = “NOF5”, “WF5” etc.

Given Multiple New subscribers <Hybrid_Subscriber> profile is <Hybrid_Offering_Code> and <Prepaid_Subscriber> profile is <Prepaid_Offering_Code>

- The function will create two subscriber profiles with provided offering codes.
- Hybrid_Subscriber/Hybrid_Offering_Code – MSISDN to be created
- Hybrid_Offering_Code/ Prepaid_Offering_Code – subscriber type, WF12/NOF7

Given Existing subscriber is <Subscriber_type> with offering code <Offering_Code>

- This will be used when you want to reuse an already existing subscriber and there is a not need to create a new subscriber.

Create CUR

- This will create a CUR profile for the subscriber you are creating .
- The billing platform and payment type are most important.
- Payment type will be the subscriber type you would like to create. Example: P for Prepaid, H for Hybrid an C for Postpaid.
- The 3 """ MUST ALWAYS BE THERE. As they currently are below – for the beginning and end of the function.
- The separator WILL ALWAYS be a coma " , " - Ensure that the last attribute doesn’t have a coma.

```
And CUR profile is set with attributes
  """
  {
   "billingplatformid": "300",
   "paymenttype": "P"
  }
  """
```

- **If anything else is required in CUR – please ensure that all CUR attributes are created as according to your project.**

FI Fusion

And Set FI specific details "FI Code" is <FICode> and schemeAgencyName is <Company>

Non Swagger services

- Eligibility test

And send nonSwagger eligibility for "<Recharge_Amount>"

And send nonSwagger eligibility for <SOID>

- FI Airtime recharges
  When nonSweggar recharge via FI Fusion recharge value of <Recharge_Amount> ZAR

- FI Bundles Purchases
  When nonSwagger purchase via fusion rest soid is <SOID> price is <Purchase_Amount> bundle size is <Bundle_Size> unit code is <unit_code> validity period is <validity_period>

Swagger Services

- Eligibility test

And send Swagger eligibility for "<Recharge_Amount>"

And send Swagger eligibility for <SOID>

- FI Airtime recharges

When Sweggar recharge via FI Fusion recharge value of <Recharge_Amount> ZAR

- FI Bundles Purchases

When Swagger purchase via fusion rest soid is <SOID> price is <Purchase_Amount> bundle size is <Bundle_Size> unit code is <unit_code> validity period is <validity_period>

Airtime balance adjust

This will only apply to either prepaid or hybrid MSISDN. It will be required when there is a need to adjust the airtime balance counter.

And Subscriber <MonetaryType_Code> balance is <Recharge_amount>

Allows for balance adjust for a specified money counter to the amount specified

- MonetaryType_Code must always be a string variable
  - example: "C_VZA_PPS_MainAccount" (Please refer to the Pricebook for different Monetary Type Codes)
- Recharge_amount - Decimal point
  - example: R4.00

Fusion purchase request

When I purchase via fusion soid is <SOID> price is <Purchase_Amount> bundle size is <Bundle_Size> unit code is <unit_code> validity period is <validity_period>

- This will send a purchase/provisioning request to CCS via Fusion. This will work for both dynamic and static bundles.
- **<unit_code>** - unit code for bundle size depending on what is being purchased/provisioned. E.g. “GB”, “MB”,”MINUTES”,”UNITS”
- **SOID** must always be a string variable
  - example: "D001"
- **Purchase_Amount** - Decimal point
  - example: R4.00
- **Bundle_size**
  - for non-decimal bundle sizes
    - example : 1MB or 1GB or 1TB
  - for decimal bundle sizes
    - example : "1.2GB"
  - for voice bundle sizes
    - example : 100MIN
  - For SMS Bundles
    - Example : 150SMS
- validity_period must always be a string variable
  - example : "7D"

- Example : "3H"

When I purchase via fusion product id is <Product_ID> price is <Purchase_Amount> bundle size is <Voice_Bundle> unit code is <unit_code>

- This will send a purchase/provisioning request to CCS via Fusion. This will work when you puurchase a FreeUnit instead of a SOID. Currently, working for Voice, Data Advance and Airtime Advance.
- Product ID is provided for in project documentation
- Example: Product_ID will be a string variable "VoiceAdvanceANAT"

When I Delete Bundle via fusion soid is <SOID> price is <Purchase_Amount> bundle size is <Data_Bundle> unit code is <unit_code> validity period is <validity_period>

- This will send a Delete/Deprovisioning the bundle specified by SOID to CCS via Fusion

Add to Bill

And Set Additional Fusion Properties <additionalstr>
When Subscriber using "Add-To-Bill" purchases bundle <SOID> price is <Purchase_Amount> bundle size is <Data_Bundle> validity period is <validity_period>

- This will do a purchase with the payload used for Add-to-Bill
- <additionalstr> will encompass the following:
  - "FI_cost:12/FormatId:B005"
    - FI_cost – is the amount you are purchasing with – to be added to bill
    - FormatId – is what you are triggering with the payload

Verify CCS

Verify Bundle Provisioned

When verifying an offer which is not prorated you can use either of the below

Then Data bundle for OfferingID <OfferingID> allocated is <Bundle_Size>

- This will verify if CCS bundle provisioning is correct or not. The bundle being verified will be identified by <OfferingID> parameter.
- NB: This is strictly for NON-PRORATED bundles.

Then Data bundle for OfferingID <OfferingID> allocated is <Bundle_Size> free unit is <FreeUnit> <Prorated>

- This will verify if CCS bundle provisioning is correct or not. The bundle being verified will be identified by <FreeUnit> parameter.
- When the bundle being verified is not prorated, then the parameter <Prorated> should be set to "NO"
- When the bundle being verified is prorated, then the parameter <Prorated> should be set to "YES"

Then SMS bundle for OfferingID <OfferingID> allocated is <Bundle_Size> free unit is *<FreeUnit> <Prorated>*

Then Voice bundle for OfferingID <OfferingID> allocated is <Bundle_Size> free unit is <FreeUnit> <Prorated>

Then Total Data bundle for OfferingID **<OfferingID>** allocated is **<Data_Bundl**e> free unit is **<FreeUnit> <Prorated>**

Then Total Data bundle for OfferingID **<OfferingID>** allocated is **<Data_Bundl**e> free unit is **<FreeUnit> <Prorated>**

For total multiple bundles

Verify Airtime Balances

Then amount <Purchase_Amount>ZAR is deducted from <Recharge_amount> <MonetaryType_Code> money counter

- This will verify if the expected amount has been deducted from CCS correct airtime counter for prepaid and hybrid plans.
- MonetaryType_Code must always be a string variable
  - example: "C_VZA_PPS_MainAccount" (Please refer to the Pricebook for different Monetary Type Codes)

Then amount <Purchase_Amount>ZAR is credited into <MonetaryType_Code> money counter

- This will verify if the expected amount has been credited into the money counter on CCS for prepaid, hybrid and postpaid plans.

NB: monetary codes are listed on the Pricebook within the Monetary Type sheet. E.g "C_VZA_PPS_MainAccount" OR "C_VZA_AirtimeAdvance"

And Verify instant Recharge limit is <InstantRechargeLimit>cents

And Verify <MoneyCounter> money counter allocation is <amount>ZAR <Prorated>

- Function is used to verify the subscription allocation for top-up plans.

Verify Bundle Allocation Expiry Period

And Expiry period is <Expiry_Date> with Offering ID <OfferingID>

- This will verify if CCS bundle expiry period is correct as per the expected expiry date which is identified by the parameter <Expiry_Date>.
- The bundle being verified will be identified by <OfferingID> parameter.
- For this these bundles, the expiry period will be defined by number of days. **<Expiry_Date>** = "30 days","14 days", "1 days" , "3 hours" etc.

And Expiry period is <Expiry_Date> with Offering ID <OfferingID> and free unit <FreeUnit> calendar month <Prorated>

- This will verify if CCS bundle expiry date is correct or not. The bundle being verified will be identified by <FreeUnit> parameter.
- At this point the assumption is that the bundle being verified is prorated, then the parameter <Prorated> should be set to "YES"
- In this case, it most likely that the expiry period is defined Calendar Month periods. **<Expiry_Date>** = "1 months", "2 months", etc.

Verify Fusion

This will verify fusion purchase responses.

This has been identified as a major course for test automation failure. **Test Analysts are encouraged to be certain if this is a requirement or not for the specific test.**

Verify Service Balances (ViewThrough)

Then verify "C_VZA_PPS_MainAccount" service balances remaining airtime

Then fusion service bundle <FreeUnit> <OfferingID> for product <Product_Name> is <Bundle_Size> <Prorated>

- This will verify if the provisioned bundle is as per the expected.
- This will verify the bundle based on the provided product name. <Product_Name> = "1.2GB - 30 Days recurring bundle"

Then fusion service validity <FreeUnit> <OfferingID> <Validity_Type> for product <Product_Name> is <Expiry_Date>

- This will verify if the expiry date is as per the expected.
- This will verify the expiry date based on the provided product name. <Validity_Type> = "validityPeriod" <Product_Name> = "1.2GB - 30 Days recurring bundle"

Then verify fusion <FreeUnit> <OfferingID> <Product_Name> <Product_desc>

- This will verify the description of the bundle is as per the expected.

Then verify fusion <Product_Name> <Product_desc>

- This will verify the Product name and description of the bundle (ONLY) is as per the expected.
- This was used for a project that did NOT have <FreeUnit>s

VoMS

When Subscriber redeems valid voucher for voucher type <Voucher_type> via IVR

When Subscriber redeems valid voucher for voucher type <Voucher_type> via USSD

When Subscriber redeems valid voucher for voucher type <Voucher_type> via SMS

- For all three VOMS redeems functions, this will perform VOMS voucher redeem/recharge based on the provided Voucher_type
- The function will automatically perform a voucher PIN generation if there is no existing and active voucher in batch stored in the database.

Product Description	Denomination (Price of Bundle)	Voucher Type
R 110 PIN vodago	110	31
R 275 PIN vodago	275	36
R 29 PIN vodago	29	16
R60 Yebo5 PIN	12	42
R49 SMS PIN	49	44
R49 SMS PIN	49	43
R25 Yebo5 PIN	5	6
P 60Min VB Pin	5	4722
P 10Min VB Pin	2	4724
P 10Min VB Pin	2	4720
P 50MB 1Hr Pin	5	4726
P 50MB 1Hr Pin	5	4727
MyMeg500DB Pin - 30 Days	79	5205
MyGig1DB Pin - 30 Days	99	5206
MONTHLY 20GB PIN	699	5211

Monthly 50MB PIN	12	5215
Monthly 325MB PIN	55	5217
Daily 60MB PIN	9	5226
Daily 100MB PIN	15	5227
Daily 250MB PIN	27	5228
Weekly 100MB PIN	17	5229
Weekly 1GB PIN	80	5232
Weekly 2GB PIN	120	5233
100MB Recur 6 Months PIN	119	5236
250MB Recur 6 Months PIN	219	5237
WEEKLY 120MB R17 PIN	17	5390
WEEKLY 2GB R99 PIN	99	5373
WEEKLY 5GB R199 PIN	199	5374
MONTHLY 2GB R149 PIN	149	5375
WEEKLY 500MB R49 PIN	49	5371
Monthly 3GB PIN	229	5208
WEEKLY 1GB R69 PIN	69	5372
Monthly 350MB	49	5194
MONTHLY 5GB PIN	349	5209
R 55 PIN vodago	55	26
Daily 20MB PIN	5	5225
WEEKLY 250MB R29 PIN	29	5370
R25 Yebo5 PIN	5	5
WA 250MB 1DAY R5 PIN	4	9484
R2.00 Electronic PIN	2	96
WA 50MB 1DAY R3 PIN	3	9482
R799/M50GB30D	799	5396
R60 Yebo5 PIN	12	41
WA 250MB 7DAY R12 PIN	12	9485
R0 Variable Pin	0	99
R 55 PIN vodago	55	25
WA 100MB 3DAY R6 PIN	6	9483
Monthly 1GB	85	5196
MONTHLY 30GB R699	699	5392
MONTHLY 4GB R249 PIN	249	5376
R 1100 PIN vodago	1100	37
Weekly 250MB PIN	35	5230
Monthly 500MB	69	5195
MONTHLY 10GB PIN	469	5210
Weekly 500MB PIN	60	5231
3GB 7Days PIN	129	5395
MONTHLY 15GB R529 PIN	529	5377
R 29 PIN vodago	29	15
Monthly 150MB PIN	29	5216
14Day 1GB PIN	100	5234
R 1100 PIN vodago	1100	38
R 275 PIN vodago	275	35
R 110 PIN vodago	110	32
R 10 Electronic PIN	10	98
MONTHLY 6GB R349 PIN	349	5391
R999/M100GB30D/TW365	999	5397
Monthly 200MB	29	5198
Hisence 30GB 30Days R599 PIN	599	6779
R35.00 WhatsApp 1GB - 30 Day	35	6782
Mobicel Epic 54GB 30Days R1199 PIN	1199	6780

Mobicel Oreo 30GB 30Days R599 PIN	599	6778
ITEL 54GB 30Days R1199 PIN	1199	6781

Purchase Voice, Data and Airtime Advance

Airtime Advance from Channel

Voice, Data and Airtime Advance from Fusion

```
When I purchase via fusion product id is <Product_ID> price is <Purchase_Amount>
bundle size is <Voice Bundle> unit code is <unit_code>
```

- This will send a purchase/provisioning request to CCS via Fusion. This will work when you purchase a FreeUnit instead of a SOID. Currently, working for Voice and Data Advance.
- Product ID is provided for in project documentation
- Example: Product_ID will be a string variable "VoiceAdvanceANAT"

```
When I purchase via fusion product id is <Product_ID> price is <Purchase_Amount>
and service fee <Servicefee> bundle size is <Voice_Bundle> unit code is
<unit_code>
```

- This will send a provision request to CCS via Fusion. Works similar to above function, but this allows you to bypass other systems and provision directly from Fusion.
- Example as above.

```
When I purchase via fusion product id is <Product_ID> price is <Purchase_Amount>
and service fee <Servicefee>
```

- This will send a provision request to CCS via Fusion for Airtime Advance. Works similar to previous Voice and Data function, but this allows you to bypass other systems and provision directly from Fusion.
- Example: Product_ID will be a string variable "AirtimeAdvance"

Verify Voice, Data Advance and Airtime Advance

Then amount <Purchase_Amount>ZAR is added into <LoanAmount>ZAR loan counter

- This will verify if the expected amount has been added to Fusion Loan amount.

Then outstanding amount <Purchase_Amount>ZAR and service <Servicefee>ZAR is deducted from recharge amount <Recharge_Amount>ZAR

- Ensure that a Recharge (Airtime Recharge NOT Account Adjust) is done before this step.
- This will verify if the Service Fee amount (currently R1.10) and the purchase amount has been deducted from the Recharge amount – i.e. The amount that was recharged with before this.

NB: These are some of the steps required. Other verifications will be required as according to your project.

Buy for another

Ensure that the 2 parties, A-Party and B-Party are created:

Given Multiple New subscribers <Subscriber_type1> profile is <Offerring_Code1> and <Subscriber_type2> profile is <Offerring_Code2>

- Ensure that the variables are included in your example.

And Subscriber <MonetaryType_Code> balance is <Recharge_amount>

- This will recharge the A-Party

Use the following line when doing a purchase for another MSISDN (B-Party):

When I buy for another via fusion soid is <SOID> price is <Purchase_Amount> bundle size is <Data_Bundle> unit code is <unit_code> validity period is <validity_period>

To check the balance for A-Party (after purchase) ensure that the deduction line comes right after the purchase line above. See **Verify Airtime Balances**

To ensure the B-Party received the purchased bundle:

And Data bundle for OfferingID <OfferingID> allocated is <Data_Bundle> is transfered to <Subscriber_type2> with free unit code <FreeUnit> <Prorated>

SMS Verification

Please NOTE: The SMS template that comes from Business will be used. This is project specific
The SMSes verified (for the time being) are ONLY FUSION.

And SMS template for SOID <SOID> is "Your {0} bundle has been provisioned and will be valid until {1,date,yyyy-MM-dd 'at' HH:mm:ss}."

- THIS IS AN EXAMPLE OF THE TEMPLATE - "Your {0} bundle has been provisioned and will be valid until {1,date,yyyy-MM-dd 'at' HH:mm:ss}."

This is the verification of the above mentioned SMS (template) sent by Fusion.

And SMS message for SOID <SOID> bundle size <Data_Bundle> and expiry date <Expiry_Date> and calendar month <Prorated> is sent

FreeChange

This function allows you to FreeChange the subscriber you created to a different package

And Subscriber is eligible to change to <toOffer>
When subscriber performs free change from product offer <Offerring_Code> to product offer <toOffer>
Then subscriber product offer is <toOffer>

- ToOffer is the NEW package that you would like to move the subscriber to.
- The last line verifies that the new package is provisioned.

Vodabucks:

To allocate or add vodabucks for the subscriber step:

```
When add {double} vodabucks for the subscriber
```

Where {double} is the amount of vodabucks you wish to add for the subscriber

To verify a deduction in vodabucks step:

```
Then verify that {double} vodabucks is deducted for the subscriber
```

Where {double} is the amount of vodabucks deducted for the subscriber

To verify vodabucks allocation step:

```
Then verify that {double} vodabucks is allocated for the subscriber
```

Where {double} is the amount of vodabucks allocated/added for the subscriber

To get vodabucks balance before a purchase step:

```
And get vodabucks balance for the subscriber
```

To bank any unbanked vodabucks step:

```
And bank vodabucks for the subscriber
```