THOMAS MORE | UNIVERSITY OF APPLIED SCIENCES

Realization automation

INTERNSHIP TELENET

MATTHIAS HEYLEN

Bachelor of Applied Computer Science

Option: Application Development

Academic Year 2024-2025
Campus Geel, Kleinhoefstraat 4, 2440 Geel

MEMBER OF ASSOCIATIE KU LEUVEN

Contents

Introduction

Automation testing plays a crucial role in modern software development, offering efficiency, reliability, and scalability in validating system functionality. During my internship at Telenet, I contributed to enhancing the quality assurance processes within the Number Porting Tribe by writing and refining automated tests. Specifically, I utilized Cucumber, a powerful tool for behavior-driven development (BDD), to create and maintain test scenarios.

Automation testing added significant value to our projects by ensuring that critical functionalities remained intact as the system evolved. It reduced the time required for manual testing, increased test coverage, and provided quick feedback loops during development. This was particularly important in the context of the fixed and mobile number porting projects, where consistent performance and accuracy were non-negotiable.

My responsibilities included writing new test cases in Cucumber and defining additional steps when required to handle specific testing scenarios. This involved understanding system requirements and ensuring that the tests were comprehensive and aligned with project needs. By doing so, I contributed to the reliability and maintainability of the system through thorough and consistent automated testing.

CNP-PORTACROSS-POSITIVE Verify Port Across from Base to Proximus when original owner is not base or temo

This is the first automation test I wrote during my internship at Telenet. The framework we use for the automation tests is called Cucumber. This allows us to write tests in readable language, making it easier for technical and non-technical stakeholders to collaborate on defining the desired behavior.

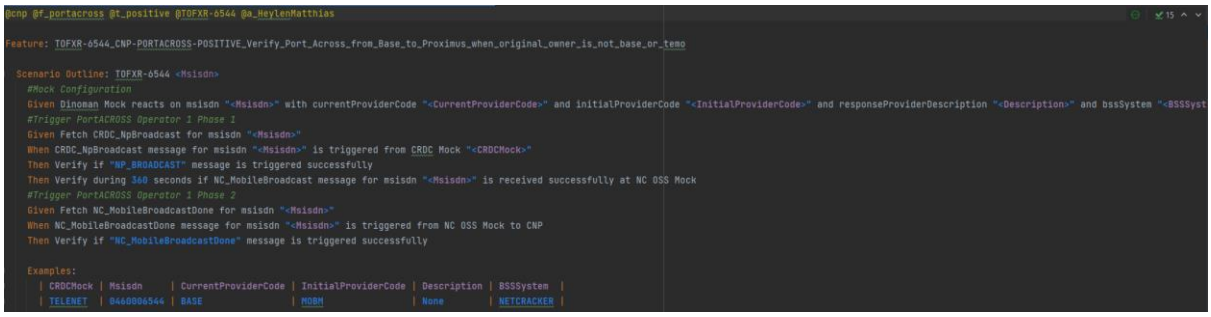
The steps are defined in a definition file, which in turn communicate via API to retrieve data. For the first test I did not need to add or modify any existing steps.

The first test I wrote is for a scenario where a port across happens. A port across message is essentially a broadcast message where a number is ported from one operator to another.

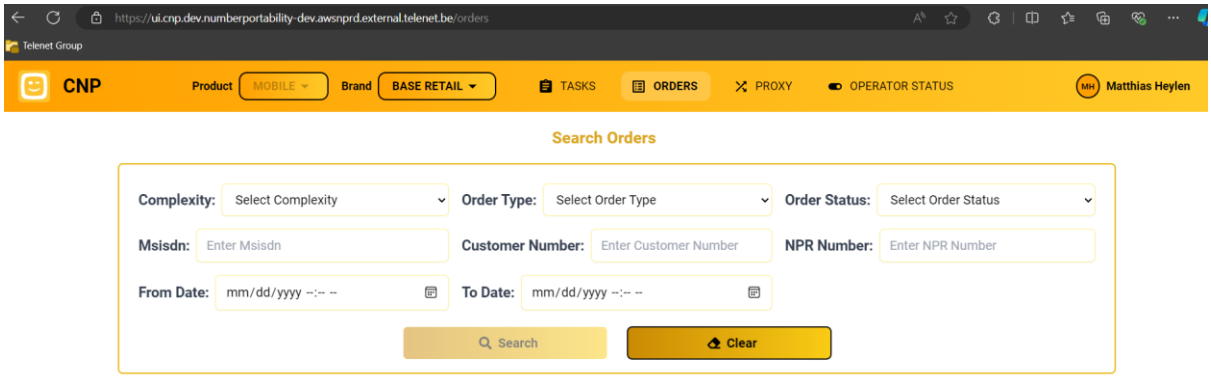
CRDC is a central system that manages both mobile and fixed-line number portability. The CRDC maintains a database that identifies which operator currently services a specific number.

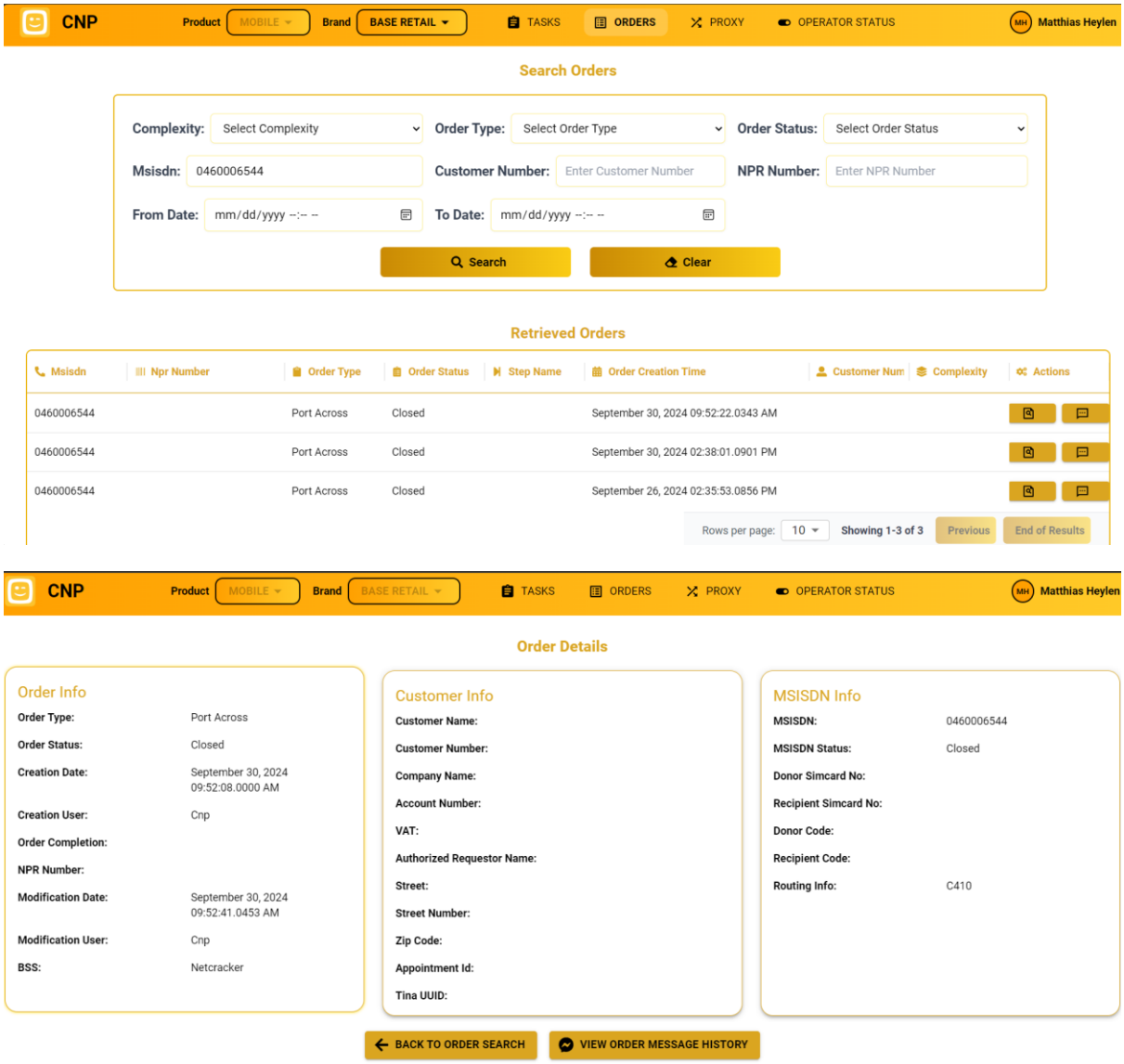
NC (Netcracker) OSS helps telecom companies manage their networks. It organizes network resources, automates task lite setting up new services, monitors network health and ensures everything runs smoothly.

Dinoman is an applications which acts as the inventory for the directory numbers.

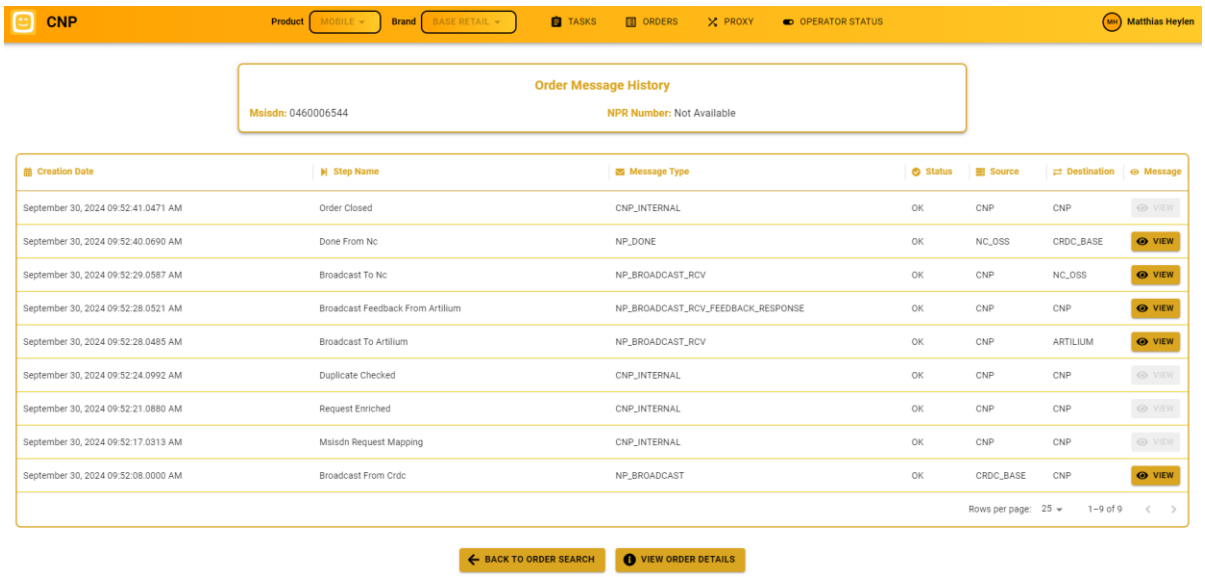


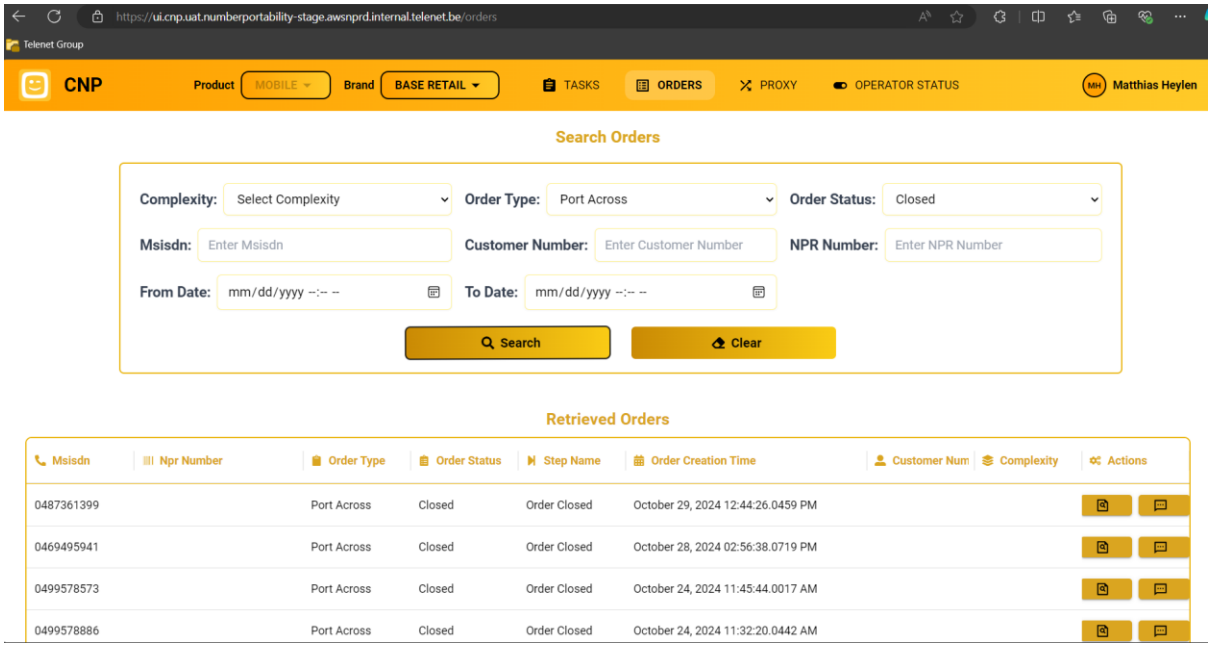
After executing the test we can see the output in the UI. We can give in a number and check the details.

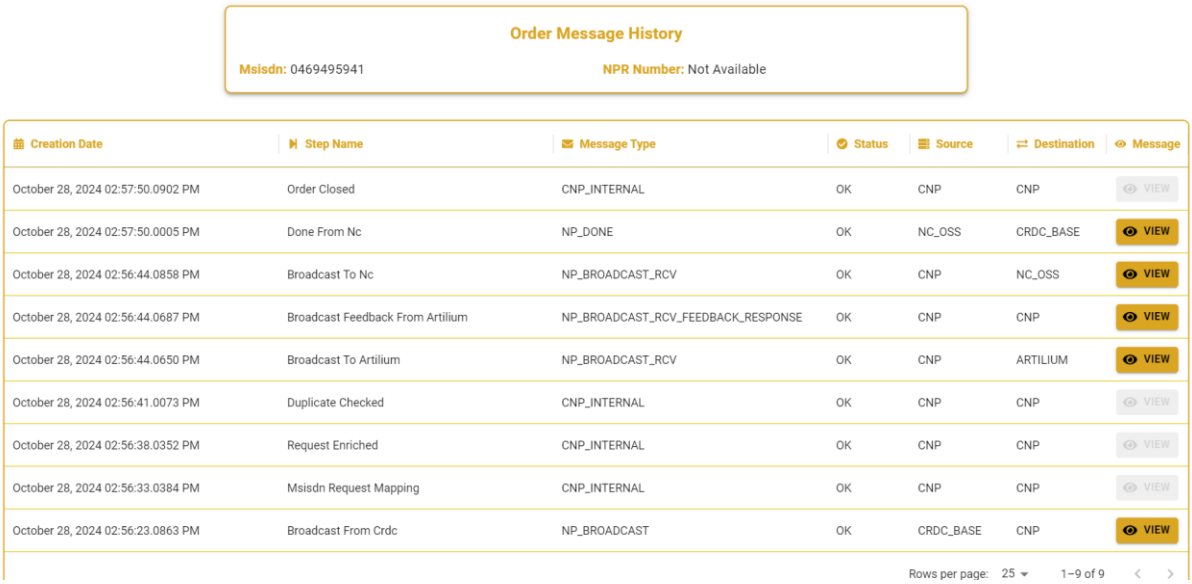


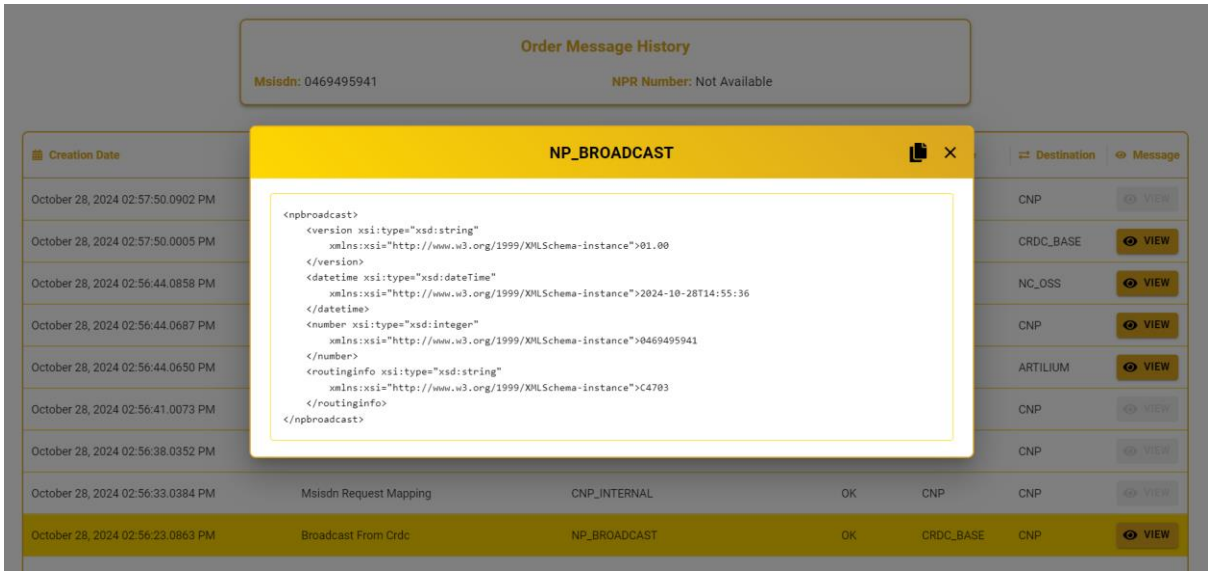


Here we can see the steps that are being performed. If the steps are executed correctly, the test is successful.

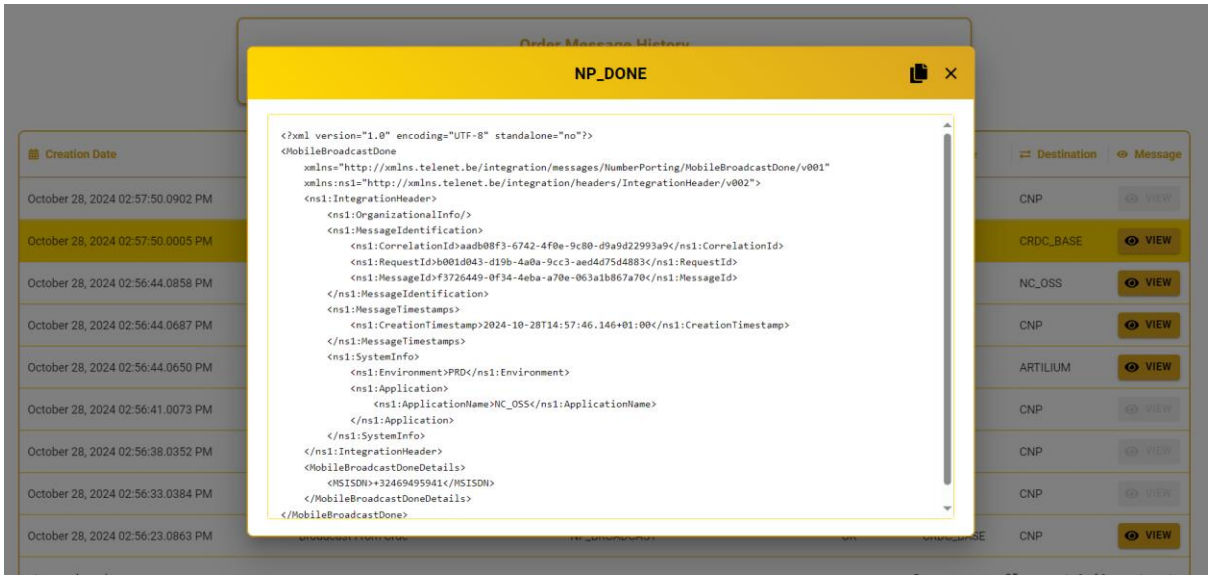



https://ui.cnp.uat.numberportability-stage.awsnprd.internal.telenet.be/orders
Telenet Group
CNP
Product
MOBILE
Brand
BASE RETAIL
TASKS
ORDERS
PROXY
OPERATOR STATUS
Matthias Heylen
Search Orders
Complexity:
Select Complexity
Order Type:
Port Across
Order Status:
Closed
Msisdn:
Enter Msisdn
Customer Number:
Enter Customer Number
NPR Number:
Enter NPR Number
From Date:
mm/dd/yyyy --:-- --
To Date:
mm/dd/yyyy --:-- --
Search
Clear
Retrieved Orders
Msisdn
Npr Number
Order Type
Order Status
Step Name
Order Creation Time
Customer Num
Complexity
Actions
0487361399
Port Across
Closed
Order Closed
October 29, 2024 12:44:26.0459 PM
0469495941
Port Across
Closed
Order Closed
October 28, 2024 02:56:38.0719 PM
0499578573
Port Across
Closed
Order Closed
October 24, 2024 11:45:44.0017 AM
0499578886
Port Across
Closed
Order Closed
October 24, 2024 11:32:20.0442 AM


Order Message History
Msisdn: 0469495941
NPR Number: Not Available
Creation Date
Step Name
Message Type
Status
Source
Destination
Message
October 28, 2024 02:57:50.0902 PM
Order Closed
CNP_INTERNAL
OK
CNP
CNP
VIEW
October 28, 2024 02:57:50.0005 PM
Done From Nc
NP_DONE
OK
NC_OSS
CRDC_BASE
VIEW
October 28, 2024 02:56:44.0858 PM
Broadcast To Nc
NP_BROADCAST_RCV
OK
CNP
NC_OSS
VIEW
October 28, 2024 02:56:44.0687 PM
Broadcast Feedback From Artilium
NP_BROADCAST_RCV_FEEDBACK_RESPONSE
OK
CNP
CNP
VIEW
October 28, 2024 02:56:44.0650 PM
Broadcast To Artilium
NP_BROADCAST_RCV
OK
CNP
ARTILIUM
VIEW
October 28, 2024 02:56:41.0073 PM
Duplicate Checked
CNP_INTERNAL
OK
CNP
CNP
VIEW
October 28, 2024 02:56:38.0352 PM
Request Enriched
CNP_INTERNAL
OK
CNP
CNP
VIEW
October 28, 2024 02:56:33.0384 PM
Msisdn Request Mapping
CNP_INTERNAL
OK
CNP
CNP
VIEW
October 28, 2024 02:56:23.0863 PM
Broadcast From Crdc
NP_BROADCAST
OK
CRDC_BASE
CNP
VIEW
Rows per page: 25
1–9 of 9

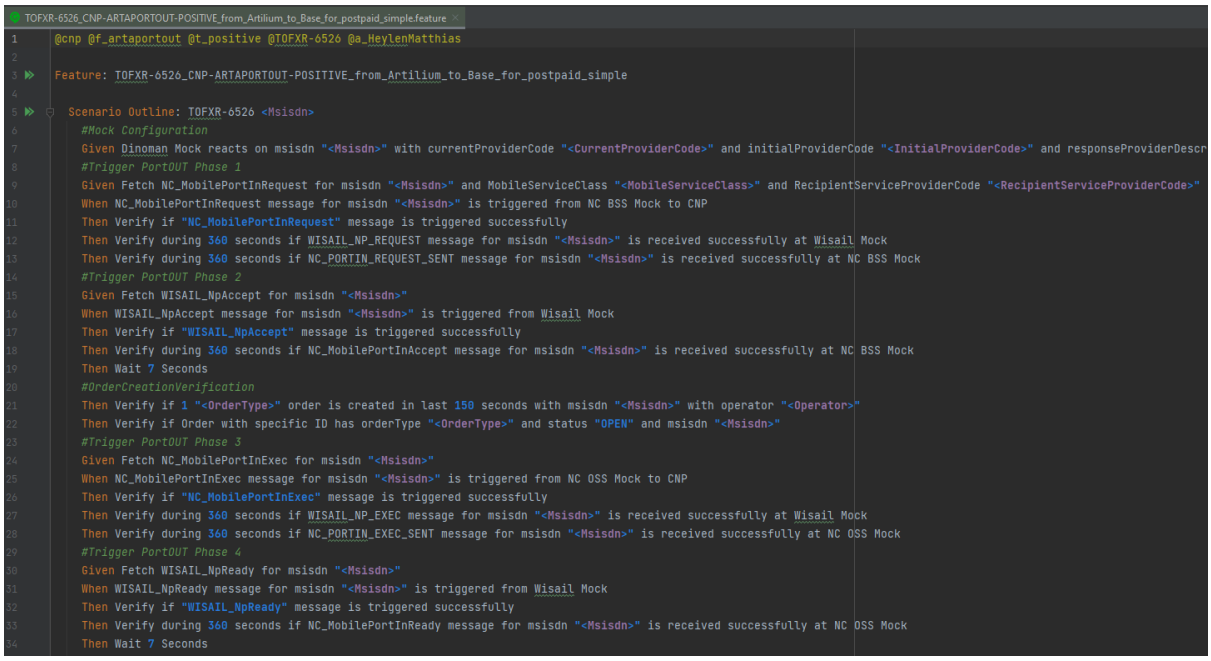

Order Message History
Msisdn: 0469495941
NPR Number: Not Available
NP_BROADCAST
<npbroadcast>
<version xsi:type="xsd:string"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">01.00
</version>
<datetime xsi:type="xsd:dateTime"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">2024-10-28T14:55:36
</datetime>
<number xsi:type="xsd:integer"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">0469495941
</number>
<routinginfo xsi:type="xsd:string"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">C4703
</routinginfo>
</npbroadcast>
October 28, 2024 02:56:33.0384 PM
Msisdn Request Mapping
CNP_INTERNAL
OK
CNP
CNP
October 28, 2024 02:56:23.0863 PM
Broadcast From Crdc
NP_BROADCAST
OK
CRDC_BASE
CNP
VIEW

For writing the automation tests it is helpful to check existing messages and see which steps they take. By doing this I know which steps are being performed and based on that information I can start writing tests.



CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Base_for_postpaid_simple

A Port-Out scenario is where number porting happens from Telenet/Base to another provider, but where we port from Base Wholesale to Base Retail is considered a special scenario as Internal Porting (Base is part of Telenet). Artilium is the support system that Base uses, while Telenet uses Netcracker.



```
#Trigger PortOUT Phase 5
Given Fetch NC_MobilePortInRFS for msisdn "<Msisdn>"
When NC_MobilePortInRFS message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
Then Verify if "NC_MobilePortInRFS" message is triggered successfully
Then Verify during 360 seconds if NC_PORTIN_RFS_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
Then Verify during 360 seconds if WISAIL_NP_BROADCAST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
Then Wait 61 Seconds
#OrderCreationVerification
Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

Examples:
  | Msisdn     | MobileServiceClass | RecipientServiceProviderCode | CurrentProviderCode | InitialProviderCode | Description | BSSSystem   | OrderType | Operator     |
  | 0460006526 | POST_PAID_SIMPLE   | BASE                         | BASE                | BASE                | None        | BASE LEGACY | PORT_IN   | BASE_RETAIL  |
```

To help construct the test I used documentation I found on Confluence, which details which steps are being taken during an Internal Porting scenario.

9. CNP - Internal porting

Created by Tiwari Vikas, last modified on Sep 17, 2024

Overview

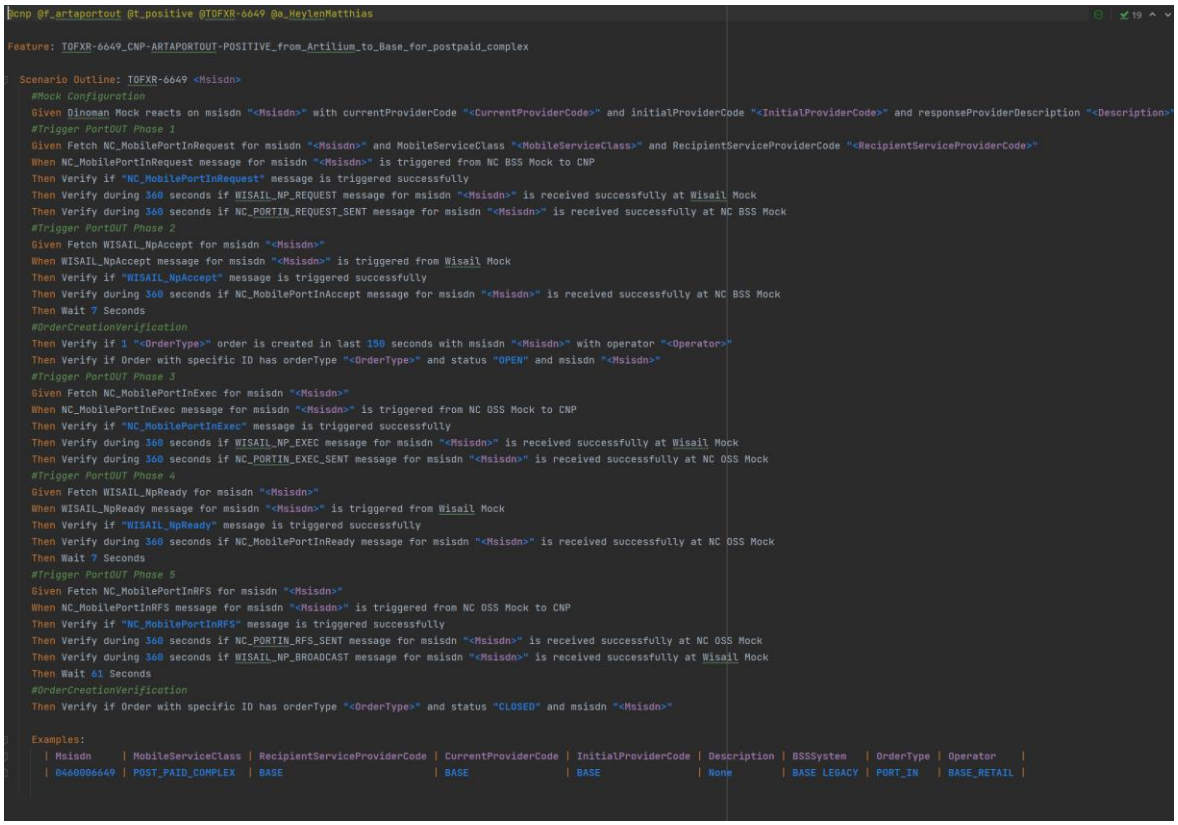
- When a number moves between BASE Wholsale & BASE retail its considered as internal porting.
- In case of internal porting we only create a port-in order for e.g. If number is moving from BASE wholesale to BASE reatil then its a port-in order for BASE retail similarly if number moves from BASE reta wholesale we will only create a port-in order for BASE wholesale.
- In case of internal porting we dont communicate with CRDC
- All the routing of messages to correct system is handled by choreographer

Message exchnage

Scenario : Number is moving from BASE Wholesale to BASE retail i.e. Port-in order for BASE retail

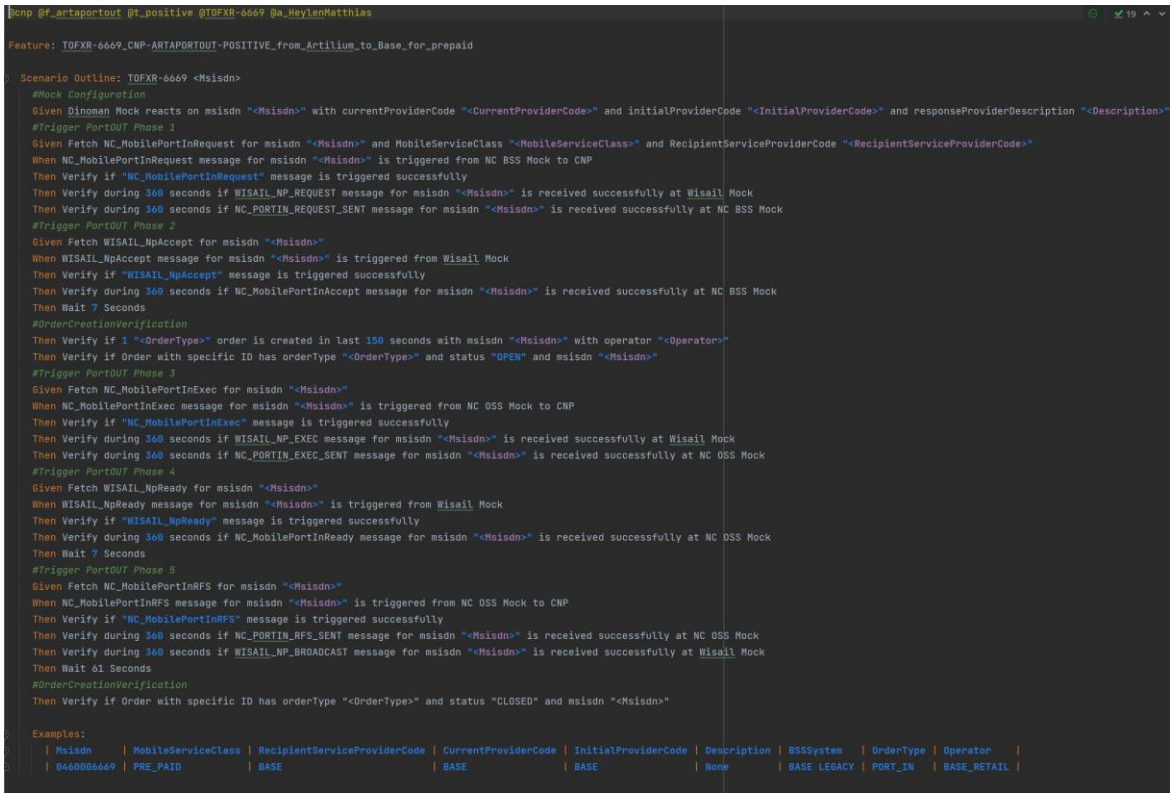
Sr no	From system	To system	Step	Message type	Sample message
1	NC	CNP	Request Received From Nc	NPR	NPR › Expand source
2	CNP	Artiliium	Request To Artilium	NP_REQUEST_RCV	NP_REQUEST_RCV › Expand source
3	Artilium	CNP	Request Feedback From Artilium	NP_REQUEST_RCV_FEEDBACK_RESPONSE	NP_REQUEST_RCV_FEEDBACK_RESPONSE › Expand source
4	CNP	NC	Request Feedback To Nc	NPR_SNT	NPR_SNT › Expand source
5	Artilium	CNP	Accept From Artilium	NP_ACCEPT	NP_ACCEPT › Expand source
6	CNP	NC	Accept To Nc	NP_ACCEPT_RCV	NP_ACCEPT_RCV › Expand source
7	NC	CNP	Exec From Nc	NP_EXEC	NP_EXEC › Expand source
8	CNP	Artilium	Exec To Artilium	NP_EXEC_RCV	NP_EXEC_RCV › Expand source
9	Artilium	CNP	Exec Feedback From Artilium	NP_EXEC_RCV_FEEDBACK_RESPONSE	NP_EXEC_RCV_FEEDBACK_RESPONSE › Expand source
10	CNP	NC	Exec Feedback To Nc	NP_EXEC_SNT	NP_EXEC_SNT › Expand source
11	Artilium	CNP	Ready From Artilium	NP_READY	NP_READY › Expand source
12	CNP	NC	Ready To Nc	NP_READY_RCV	NP_READY_RCV › Expand source
13	NC	CNP	Rfs From Nc	NP_RFS	NP_RFS › Expand source
14	CNP	Artilium	Broadcast To Artilium	NP_BROADCAST_RCV	NP_BROADCAST_RCV › Expand source
15	Artilium	CNP	Broadcast Feedback From Artilium	NP_BROADCAST_RCV_FEEDBACK_RESPONSE	NP_BROADCAST_RCV_FEEDBACK_RESPONSE › Expand source
16	CNP	NC	Broadcast Feedback To Nc	NP_RFS_SNT	

CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Base_for_postpaid_complex



```
@cnp @f_artaportout @t_positive @TOFXR-6649 @a_HeylenMatthias

Feature: TOFXR-6649_CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Base_for_postpaid_complex

  Scenario Outline: TOFXR-6649 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>"
    #Trigger PortOUT Phase 1
    Given Fetch NC_MobilePortInRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and RecipientServiceProviderCode "<RecipientServiceProviderCode>"
    When NC_MobilePortInRequest message for msisdn "<Msisdn>" is triggered from NC BSS Mock to CNP
    Then Verify if "NC_MobilePortInRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_REQUEST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTIN_REQUEST_SENT message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    #Trigger PortOUT Phase 2
    Given Fetch WISAIL_NpAccept for msisdn "<Msisdn>"
    When WISAIL_NpAccept message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpAccept" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortInAccept message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortOUT Phase 3
    Given Fetch NC_MobilePortInExec for msisdn "<Msisdn>"
    When NC_MobilePortInExec message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobilePortInExec" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_EXEC message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTIN_EXEC_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    #Trigger PortOUT Phase 4
    Given Fetch WISAIL_NpReady for msisdn "<Msisdn>"
    When WISAIL_NpReady message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpReady" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortInReady message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 5
    Given Fetch NC_MobilePortInRFS for msisdn "<Msisdn>"
    When NC_MobilePortInRFS message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobilePortInRFS" message is triggered successfully
    Then Verify during 360 seconds if NC_PORTIN_RFS_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Verify during 360 seconds if WISAIL_NP_BROADCAST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 61 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | MobileServiceClass | RecipientServiceProviderCode | CurrentProviderCode | InitialProviderCode | Description | BSSSystem   | OrderType | Operator    |
      | 0460006649 | POST_PAID_COMPLEX  | BASE                         | BASE                | BASE                | None        | BASE LEGACY | PORT_IN   | BASE_RETAIL |
```

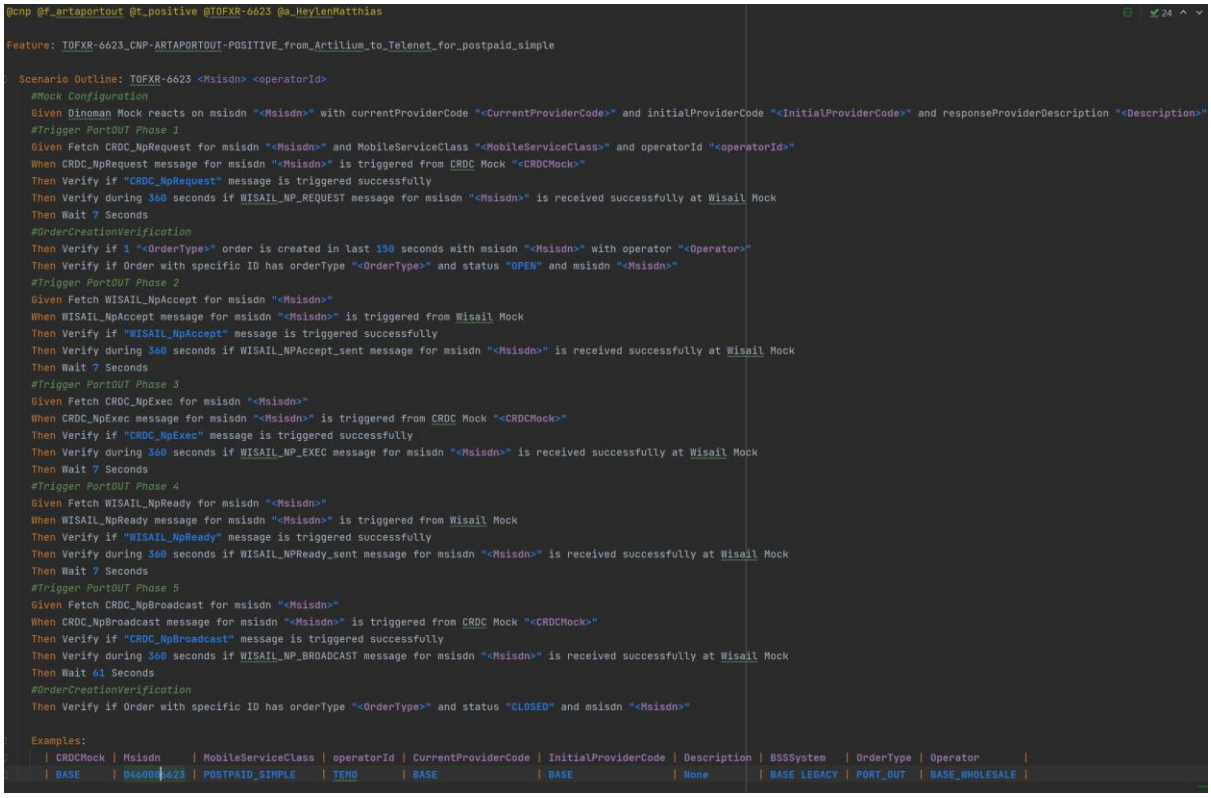
CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Base_for_prepaid



```
@cnp @f_artaportout @t_positive @TOFXR-6669 @a_HeylenMatthias

Feature: TOFXR-6669_CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Base_for_prepaid

  Scenario Outline: TOFXR-6669 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>"
    #Trigger PortOUT Phase 1
    Given Fetch NC_MobilePortInRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and RecipientServiceProviderCode "<RecipientServiceProviderCode>"
    When NC_MobilePortInRequest message for msisdn "<Msisdn>" is triggered from NC BSS Mock to CNP
    Then Verify if "NC_MobilePortInRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_REQUEST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTIN_REQUEST_SENT message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    #Trigger PortOUT Phase 2
    Given Fetch WISAIL_NpAccept for msisdn "<Msisdn>"
    When WISAIL_NpAccept message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpAccept" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortInAccept message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortOUT Phase 3
    Given Fetch NC_MobilePortInExec for msisdn "<Msisdn>"
    When NC_MobilePortInExec message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobilePortInExec" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_EXEC message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTIN_EXEC_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    #Trigger PortOUT Phase 4
    Given Fetch WISAIL_NpReady for msisdn "<Msisdn>"
    When WISAIL_NpReady message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpReady" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortInReady message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 5
    Given Fetch NC_MobilePortInRFS for msisdn "<Msisdn>"
    When NC_MobilePortInRFS message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobilePortInRFS" message is triggered successfully
    Then Verify during 360 seconds if NC_PORTIN_RFS_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Verify during 360 seconds if WISAIL_NP_BROADCAST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 61 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | MobileServiceClass | RecipientServiceProviderCode | CurrentProviderCode | InitialProviderCode | Description | BSSSystem   | OrderType | Operator    |
      | 0460006669 | PRE_PAID           | BASE                         | BASE                | BASE                | None        | BASE LEGACY | PORT_IN   | BASE_RETAIL |
```

CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Telenet_for_postpaid_simple

Compared to the previous tests, a Port-Out between Base and Telenet is not seen as internal porting.



```
@cnp @f_artaportout @t_positive @TOFXR-6623 @a_HeylenMatthias

Feature: TOFXR-6623_CNP-ARTAPORTOUT-POSITIVE_from_Artilium_to_Telenet_for_postpaid_simple

  Scenario Outline: TOFXR-6623 <Msisdn> <operatorId>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>"
    #Trigger PortOUT Phase 1
    Given Fetch CRDC_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>"
    When CRDC_NpRequest message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
    Then Verify if "CRDC_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_REQUEST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortOUT Phase 2
    Given Fetch WISAIL_NpAccept for msisdn "<Msisdn>"
    When WISAIL_NpAccept message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpAccept" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPAccept_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 3
    Given Fetch CRDC_NpExec for msisdn "<Msisdn>"
    When CRDC_NpExec message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
    Then Verify if "CRDC_NpExec" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_EXEC message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 4
    Given Fetch WISAIL_NpReady for msisdn "<Msisdn>"
    When WISAIL_NpReady message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpReady" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPReady_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 5
    Given Fetch CRDC_NpBroadcast for msisdn "<Msisdn>"
    When CRDC_NpBroadcast message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
    Then Verify if "CRDC_NpBroadcast" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_BROADCAST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 61 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | CRDCMock | Msisdn     | MobileServiceClass | operatorId | CurrentProviderCode | InitialProviderCode | Description | BSSSystem   | OrderType | Operator       |
      | BASE     | 0460006623 | POSTPAID_SIMPLE    | TEMO       | BASE                | BASE                | None        | BASE LEGACY | PORT_OUT  | BASE_WHOLESALE |
```

CNP-FNPBROADCAST-POSITIVE_Process_nprfsbroadcast_Message

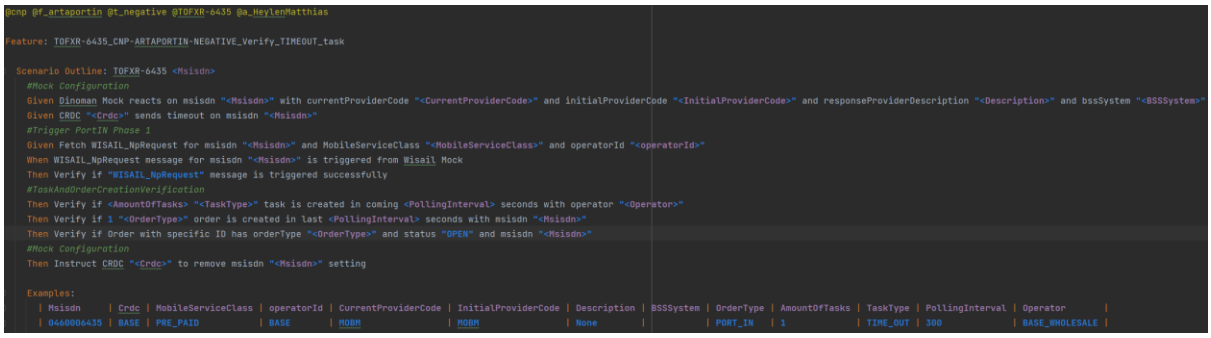
A test where we check if the broadcast messaged is being triggered successfully.

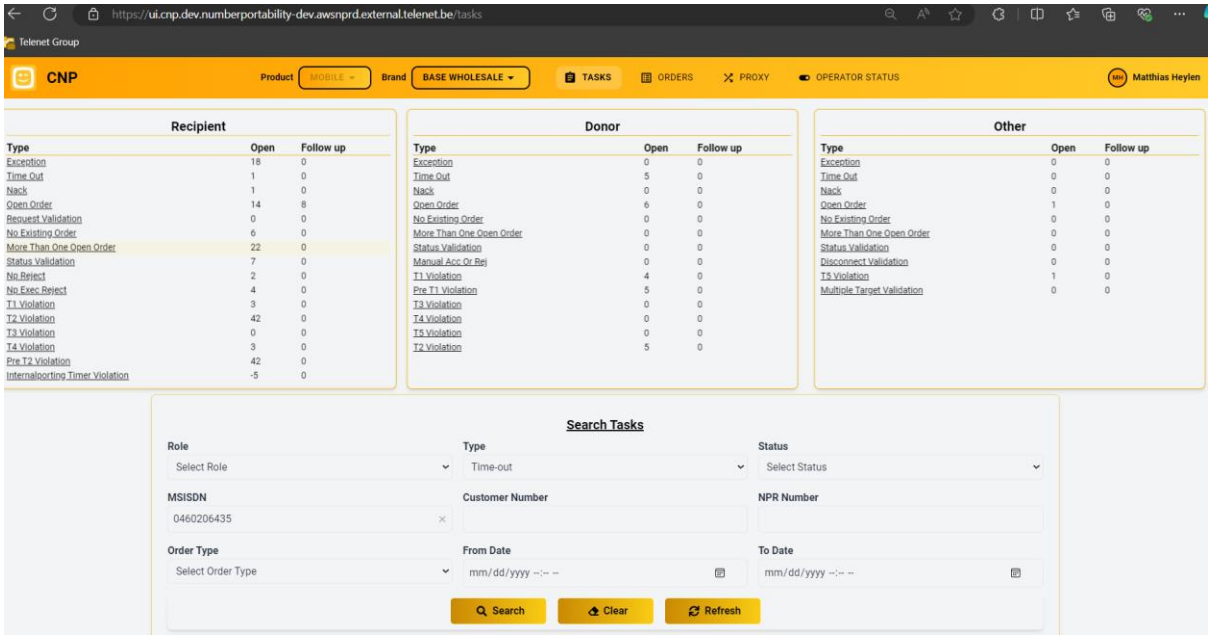
```
@cnp @f_fnpbroadcast @t_positive @TOFXR-6596 @a_HeylenMatthias

Feature: TOFXR-6596_CNP-FNPBROADCAST-POSITIVE_Process_nprfsbroadcast_Message

  Scenario Outline: TOFXR-6596 <Msisdn>
    #Trigger NP Rfs Broadcast
    Given Fetch CRDC_NpRfsBroadcast for msisdn "<Msisdn>"
    When CRDC_NpRfsBroadcast message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
    Then Verify if "CRDC_NpRfsBroadcast" message is triggered successfully

    Examples:
      | CRDCMock | Msisdn     |
      | TELENET  | 0460006596 |
      | BASE     | 0460106596 |
```

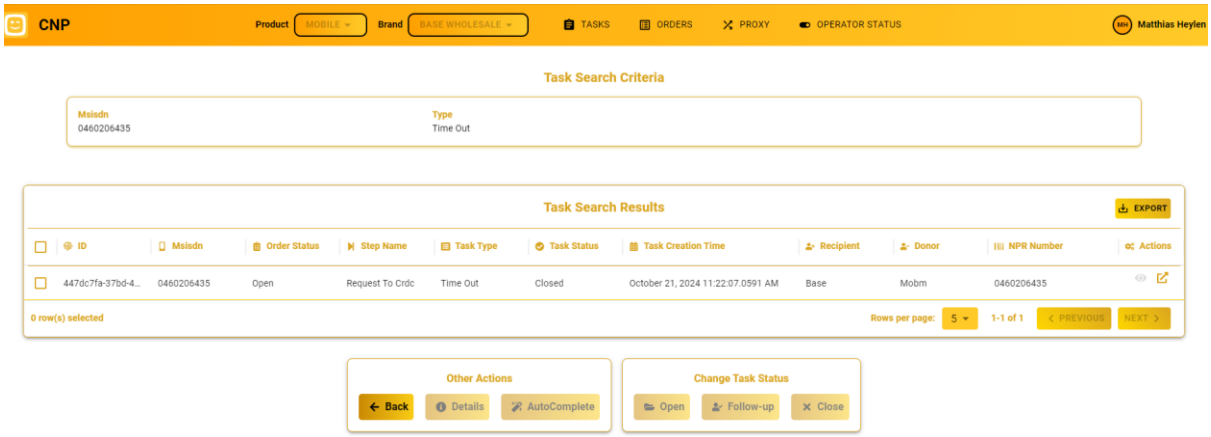
CNP-ARTAPORTIN-NEGATIVE Verify TIMEOUT task

This was my first negative test I had to make. Compared to a Positive test, a Negative test fails somewhere. In this test we remove the msisdn setting so we can see if a timeout is thrown.

To see if a timeout occurs, we can check under “tasks” in the UI and see if a timeout task is created for our msisdn.

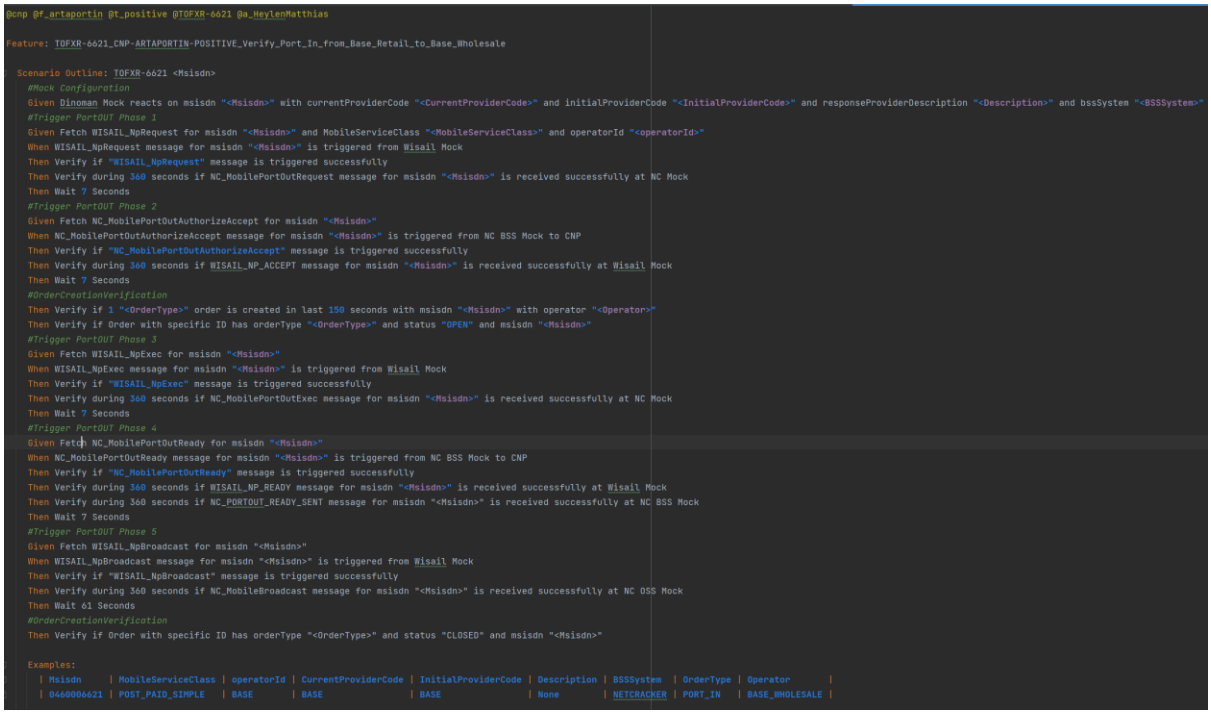






CNP-ARTAPORTIN-POSITIVE Verify Port In from Base Retail to Base Wholesale

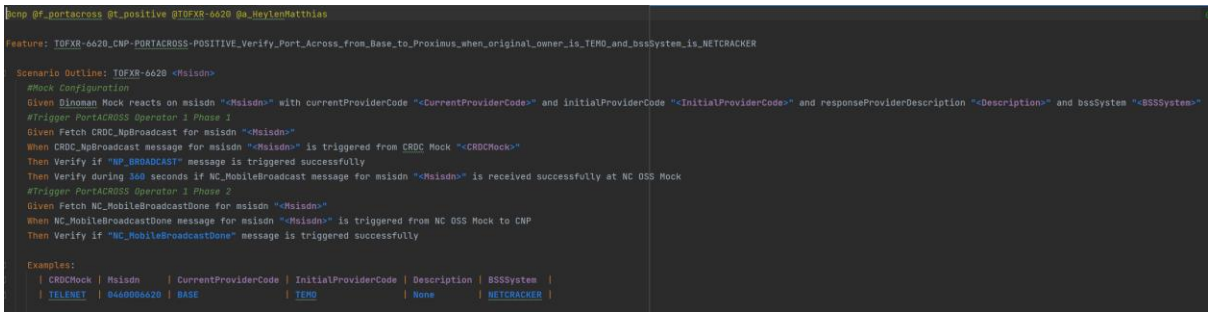
A Port-In scenario where a Port In happens between Base Retail and Base Wholesale.



```
@cnp @f_artaportin @t_positive @TOFXR-6621 @a_HeylenMatthias

Feature: TOFXR-6621_CNP-ARTAPORTIN-POSITIVE_Verify_Port_In_from_Base_Retail_to_Base_Wholesale

  Scenario Outline: TOFXR-6621 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortOUT Phase 1
    Given Fetch WISAIL_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>"
    When WISAIL_NpRequest message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortOutRequest message for msisdn "<Msisdn>" is received successfully at NC Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 2
    Given Fetch NC_MobilePortOutAuthorizeAccept for msisdn "<Msisdn>"
    When NC_MobilePortOutAuthorizeAccept message for msisdn "<Msisdn>" is triggered from NC BSS Mock to CNP
    Then Verify if "NC_MobilePortOutAuthorizeAccept" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_ACCEPT message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortOUT Phase 3
    Given Fetch WISAIL_NpExec for msisdn "<Msisdn>"
    When WISAIL_NpExec message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpExec" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortOutExec message for msisdn "<Msisdn>" is received successfully at NC Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 4
    Given Fetch NC_MobilePortOutReady for msisdn "<Msisdn>"
    When NC_MobilePortOutReady message for msisdn "<Msisdn>" is triggered from NC BSS Mock to CNP
    Then Verify if "NC_MobilePortOutReady" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_READY message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTOUT_READY_SENT message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 5
    Given Fetch WISAIL_NpBroadcast for msisdn "<Msisdn>"
    When WISAIL_NpBroadcast message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpBroadcast" message is triggered successfully
    Then Verify during 360 seconds if NC_MobileBroadcast message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Wait 61 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | MobileServiceClass | operatorId | CurrentProviderCode | InitialProviderCode | Description | BSSSystem  | OrderType | Operator       |
      | 0460006621 | POST_PAID_SIMPLE   | BASE       | BASE                | BASE                | None        | NETCRACKER | PORT_IN   | BASE_WHOLESALE |
```

CNP-PORTACROSS-POSITIVE_Verify Port Across from Base to Proximus when original owner is TEMO and bssSystem is NETCRACKER

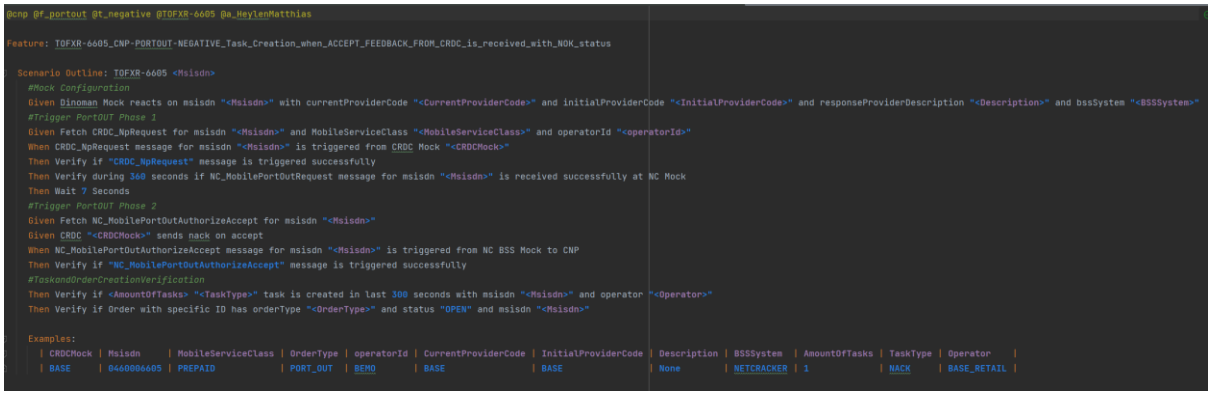
A Port-Across scenario with number porting between Base and Proximus, where the original owner is Telenet.



```
@cnp @f_portacross @t_positive @TOFXR-6620 @a_HeylenMatthias

Feature: TOFXR-6620_CNP-PORTACROSS-POSITIVE_Verify_Port_Across_from_Base_to_Proximus_when_original_owner_is_TEMO_and_bssSystem_is_NETCRACKER

  Scenario Outline: TOFXR-6620 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortACROSS Operator 1 Phase 1
    Given Fetch CRDC_NpBroadcast for msisdn "<Msisdn>"
    When CRDC_NpBroadcast message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
    Then Verify if "NP_BROADCAST" message is triggered successfully
    Then Verify during 360 seconds if NC_MobileBroadcast message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    #Trigger PortACROSS Operator 1 Phase 2
    Given Fetch NC_MobileBroadcastDone for msisdn "<Msisdn>"
    When NC_MobileBroadcastDone message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobileBroadcastDone" message is triggered successfully

    Examples:
      | CRDCMock | Msisdn     | CurrentProviderCode | InitialProviderCode | Description | BSSSystem  |
      | TELENET  | 0460006620 | BASE                | TEMO                | None        | NETCRACKER |
```

CNP-PORTOUT-NEGATIVE Task Creation when ACCEPT_FEEDBACK_FROM_CRDC is received with NOK status

For this test I had to create a new step in the definition class. In the @given we define how the step should be used/named in the test class. In the step definition class we communicate with the API to send a nack so we can receive a NOK status during the ACCEPT_FEEDBACK_FROM_CRDC step in our messaging process.



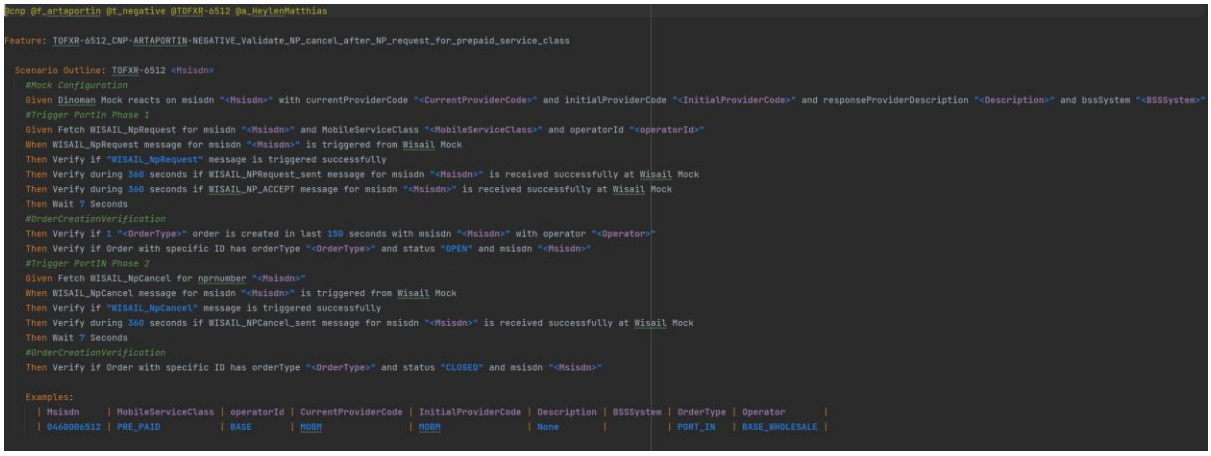
```
@cnp @f_portout @t_negative @TOFXR-6605 @a_HeylenMatthias

Feature: TOFXR-6605_CNP-PORTOUT-NEGATIVE_Task_Creation_when_ACCEPT_FEEDBACK_FROM_CRDC_is_received_with_NOK_status

  Scenario Outline: TOFXR-6605 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortOUT Phase 1
    Given Fetch CRDC_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>"
    When CRDC_NpRequest message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
    Then Verify if "CRDC_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortOutRequest message for msisdn "<Msisdn>" is received successfully at NC Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 2
    Given Fetch NC_MobilePortOutAuthorizeAccept for msisdn "<Msisdn>"
    Given CRDC "<CRDCMock>" sends nack on accept
    When NC_MobilePortOutAuthorizeAccept message for msisdn "<Msisdn>" is triggered from NC BSS Mock to CNP
    Then Verify if "NC_MobilePortOutAuthorizeAccept" message is triggered successfully
    #TaskandOrderCreationVerification
    Then Verify if <AmountOfTasks> "<TaskType>" task is created in last 300 seconds with msisdn "<Msisdn>" and operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"

    Examples:
      | CRDCMock | Msisdn     | MobileServiceClass | OrderType | operatorId | CurrentProviderCode | InitialProviderCode | Description | BSSSystem  | AmountOfTasks | TaskType | Operator     |
      | BASE     | 0460006605 | PREPAID            | PORT_OUT  | BEMO       | BASE                | BASE                | None        | NETCRACKER | 1             | NACK     | BASE_RETAIL  |
```

```
1 usage   mheylen1 +1
@Given("CRDC {string} sends nack on accept")
public void crdcSendNackForAccept(String crdc) throws Exception {
    ExtentTest test = world.getExtent().createTest(String.format("Initiate %s admin for accept - nack", crdc));
    try {
        String nprNumber = configuration.get("nprNumber");
        world.setResponse(npmanApi.sendOnReceiveCommandsNackOnNpAccept(crdc, nprNumber));
    } catch (Exception e) {
        world.setTestStatus(false);
        test.log(Status.FAIL,  details: "Failed Due to (" + e.getMessage() + ")");
        throw e;
    }
}
```

```
1 usage   mheylen1
public CloseableHttpResponse sendOnReceiveCommandsNackOnNpAccept(String crdc, String nprNumber) throws IOException {

    String admincrdcMockEndpoint = getCRDCEndpointByOperator(crdc);
    if (log.isDebugEnabled()) {
        log.debug("Admin CRDC endpoint: " + admincrdcMockEndpoint);
    }
    HttpPost postRequest = new HttpPost( uri: admincrdcMockEndpoint + "/admin/on-receive-commands/sendnack");
    String timeOut = String.format("{\"on_npr_number\":\"%s\"}", nprNumber);

    postRequest.setEntity(new StringEntity(timeOut, ContentType.APPLICATION_JSON));
    CloseableHttpResponse response = httpClient.execute(postRequest);
    log.info(
            "CRDC MOCK - " + nprNumber + " - Initiated admin crdc on sending nack: Status Code [{}]",
            response.getCode());
    return response;
}
```

CNP-ARTAPORTIN-NEGATIVE Validate NP cancel after NP request for prepaid service class

Another scenario with no existing steps or logic to execute or validate a number porting cancel in a Port In Scenario. First, we trigger the cancel for the correct nprnumber (which is just the msisdn in this scenario). After we triggered the cancel, we validate if the cancel was executed for the correct nprnumber/msisdn.



```
@cnp @f_artaportin @t_negative @TOFXR-6512 @a_HeylenMatthias

Feature: TOFXR-6512_CNP-ARTAPORTIN-NEGATIVE_Validate_NP_cancel_after_NP_request_for_prepaid_service_class

  Scenario Outline: TOFXR-6512 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortIn Phase 1
    Given Fetch WISAIL_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>"
    When WISAIL_NpRequest message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPRequest_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if WISAIL_NP_ACCEPT message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortIN Phase 2
    Given Fetch WISAIL_NpCancel for nprnumber "<Msisdn>"
    When WISAIL_NpCancel message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpCancel" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPCancel_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | MobileServiceClass | operatorId | CurrentProviderCode | InitialProviderCode | Description | BSSSystem | OrderType | Operator      |
      | 0460006512 | PRE_PAID           | BASE       | MOBM                | MOBM                | None        |           | PORT_IN   | BASE_WHOLESALE |
```

```
1 usage   mheylen1
@Given("Fetch WISAIL_NpCancel for nprnumber {string}")
public void fetch_WISAIL_NpCancel(String NprNumber) {
    Map<String, String> testStrings = generateFetchStrings( mockingName: "WISAIL_MOCK",  requestName: "WISAIL_NpCancel", NprNumber);
    String testName = testStrings.get("testName");
    String testStatusMessagePass = testStrings.get("testStatusMessagePass");
    String testStatusMessageFail = testStrings.get("testStatusMessageFail");
    ExtentTest test = world.getExtent().createTest(testName);
    try {
        world.setWISAIL_NpCancel(npmanApi.fetch_WISAIL_NpCancel(NprNumber));
        test.log(Status.PASS, testStatusMessagePass);
    } catch (Exception e) {
        test.log(Status.FAIL,  details: testStatusMessageFail + " (" + e.getMessage() + ")");
        world.setTestStatus(false);
        throw new RuntimeException(e);
    }
}
```

```
public String fetch_WISAIL_NpCancel(String NprNumber) throws IOException {
    fileName = "WISAIL_NpCancel.json";
    ClassPathResource resource = new ClassPathResource("Generic_Messages/" + fileName);
    String jsonContent = FileUtils.readFileToString(resource.getFile(), StandardCharsets.UTF_8);
    jsonContent = jsonContent.replace( target: "$nprNumber", NprNumber);
    return jsonContent;
}
```

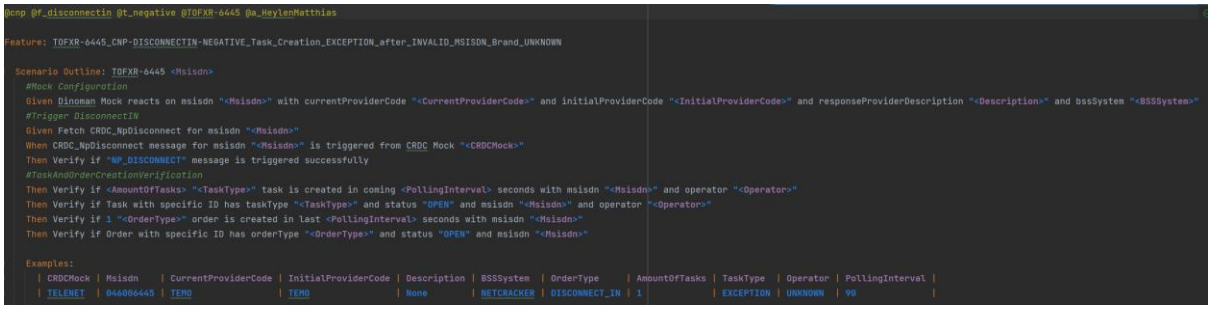
```
1 usage  mheylen1 +3
@Then("Verify during {int} seconds if WISAIL_NPCancel_sent message for msisdn {string} is received successfully at Wisail Mock")
public void verify_reception_of_WISAIL_NP_CANCEL_SENT(int pollDuration, String MSISDN) {
    Map<String, String> testStrings = generateVerifyReceptionStrings( mockingName: "WISAIL_MOCK",  requestName: "WISAIL_NPCancel_sent", MSISDN);
    String testName = testStrings.get("testName");
    String assertName = testStrings.get("assertName");
    String testStatusMessagePass = testStrings.get("testStatusMessagePass");
    String testStatusMessageFail = testStrings.get("testStatusMessageFail");
    ExtentTest test = world.getExtent().createTest(testName);
    try {
        Awaitility.await().atMost(Duration.ofSeconds(pollDuration)).pollDelay(Duration.ofSeconds(3)).pollInterval(Duration.ofSeconds(7)).un
            String res = npmanApi.retrieve_WISAIL_NPCancel_sent_For_MSISDN(MSISDN, test);
            String res1 = res.split( regex: ";")[1];
            Assert.assertEquals(res1, assertName);
        });
    } catch (AssertionError | ConditionTimeoutException e) {
        test.log(Status.FAIL,  details: testStatusMessageFail + " (" + e.getMessage() + ")");
        world.setTestStatus(false);
        throw e;
    }
    test.log(Status.PASS, testStatusMessagePass);
    world.setTestStatus(true);
}
```

```
1 usage  mheylen1 +2
public String retrieve_WISAIL_NPCancel_sent_For_MSISDN(String MSISDN, ExtentTest test)
        throws Exception {
    // Log Definitions
    final String mockingName = "WISAIL_MOCK";
    final String requestName = "WISAIL_NPCancel_sent";
    final String testStatusMessageInit = mockingName + " - Searching " + requestName;
    final String testStatusMessagePass = mockingName + " - Found " + requestName;
    final String testStatusMessageNFnd = mockingName + " - NOT Found " + requestName;
    // Search Request at Mock
    String response = getWisailMessages();
    log.info(
            String.format("%s for MSISDN: %s", testStatusMessageInit, MSISDN));
    Document parsed = dBuilder.parse(IOUtils.toInputStream(response, StandardCharsets.UTF_8));
    String successXpath1 = String.format(
            "//providefeedbackRequest[number='%s' and feedback[feedbackstring='%s']]", MSISDN, "NPCancel sent");
    Node successNode1 = (Node)
            XPathFactory.newInstance().newXPath().compile(successXpath1).evaluate(parsed, XPathConstants.NODE);
    logExtracted(successNode1);
    // Evaluate Search Result
    String testStatus = null;
    String testStatusMessage = null;
    if (successNode1 != null) {
        testStatus = "PASS";
        testStatusMessage = testStatusMessagePass;
        test.log(Status.PASS, testStatusMessage);
```

```
    } else {
        testStatus = "FAIL";
        testStatusMessage = testStatusMessageNFnd;
        test.log(Status.FAIL, testStatusMessage);
    }
    log.info(String.format("%s for MSISDN: %s", testStatusMessage, MSISDN));
    return String.format("%s;%s", testStatus, testStatusMessage);
}
```

CNP-DISCONNECTIN-NEGATIVE_Task_Creation_EXCEPTION_after_INVALID_MSISDN_Brand_UNKNOWN

In this test we use an invalid msisdn to see if an Exception task is being created.



```
@cnp @f_disconnectin @t_negative @TOFXR-6445 @a_HeylenMatthias

Feature: TOFXR-6445_CNP-DISCONNECTIN-NEGATIVE_Task_Creation_EXCEPTION_after_INVALID_MSISDN_Brand_UNKNOWN

 Scenario Outline: TOFXR-6445 <Msisdn>
   #Mock Configuration
   Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
   #Trigger DisconnectIN
   Given Fetch CRDC_NpDisconnect for msisdn "<Msisdn>"
   When CRDC_NpDisconnect message for msisdn "<Msisdn>" is triggered from CRDC Mock "<CRDCMock>"
   Then Verify if "NP_DISCONNECT" message is triggered successfully
   #TaskAndOrderCreationVerification
   Then Verify if <AmountOfTasks> "<TaskType>" task is created in coming <PollingInterval> seconds with msisdn "<Msisdn>" and operator "<Operator>"
   Then Verify if Task with specific ID has taskType "<TaskType>" and status "OPEN" and msisdn "<Msisdn>" and operator "<Operator>"
   Then Verify if 1 "<OrderType>" order is created in last <PollingInterval> seconds with msisdn "<Msisdn>"
   Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"

   Examples:
     | CRDCMock | Msisdn    | CurrentProviderCode | InitialProviderCode | Description | BSSSystem  | OrderType     | AmountOfTasks | TaskType  | Operator | PollingInterval |
     | TELENET  | 046006445 | TEMO                | TEMO                | None        | NETCRACKER | DISCONNECT_IN | 1             | EXCEPTION | UNKNOWN  | 90              |
```

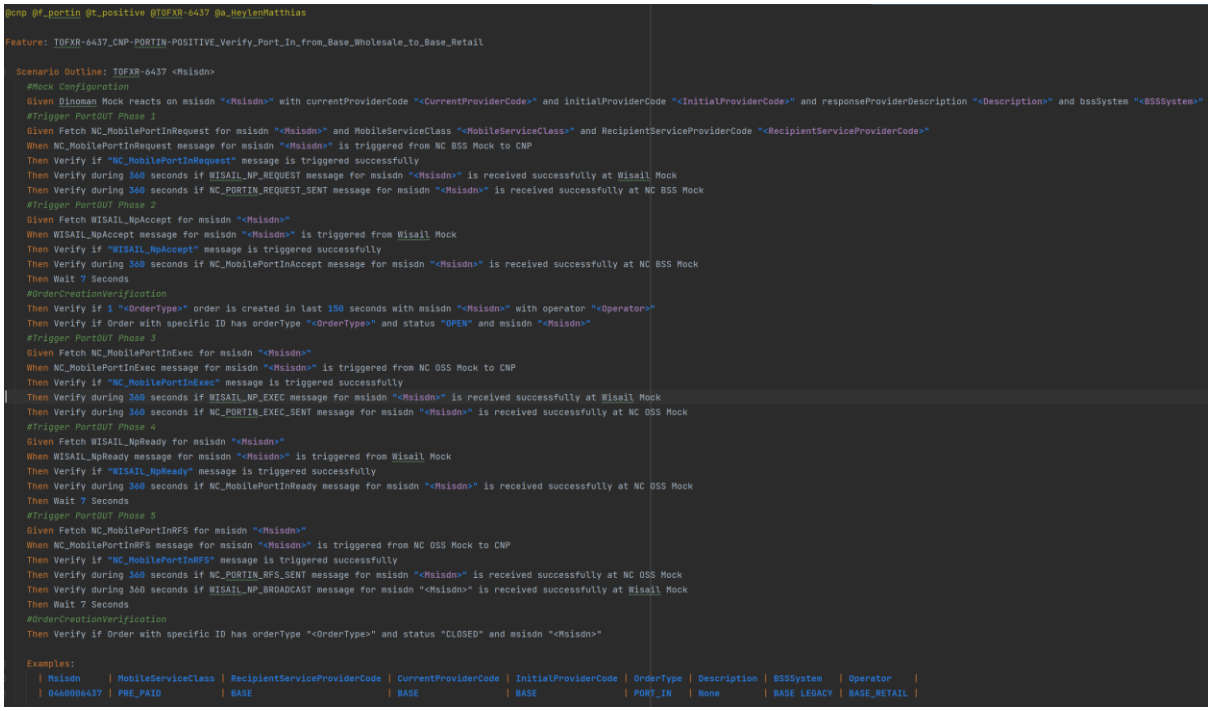
CNP-ARTAPORTIN-NEGATIVE Validate No existing order present for MNP item

In this test we see if a No Existing Order task is being created for an order that does not exist.

```
@cnp @f_artaportin @t_negative @TOFXR-6508 @a_HeylenMatthias

Feature: TOFXR-6508_CNP-ARTAPORTIN-NEGATIVE_Validate_No_existing_order_present_for_MNP_item

 Scenario Outline: TOFXR-6508 <Msisdn>
   #Trigger PortIN Exec
   Given Fetch WISAIL_NpExec for msisdn "<Msisdn>"
   When WISAIL_NpExec message for msisdn "<Msisdn>" is triggered from Wisail Mock
   Then Wait 10 Seconds
   #TaskAndOrderCreationVerification
   Then Verify if <AmountOfTasks> "<TaskType>" task is created in last 300 seconds with msisdn "<Msisdn>" and operator "<Operator>"
   Then Verify if Order with specific ID has orderType "<OrderType>" and status "FAILED" and msisdn "<Msisdn>"

   Examples:
     | Msisdn     | AmountOfTasks | TaskType          | OrderType | Operator       |
     | 0460006508 | 1             | NO_EXISTING_ORDER | PORT_IN   | BASE_WHOLESALE |
```

CNP-PORTIN-POSITIVE Verify Port In from Base Wholesale to Base Retail

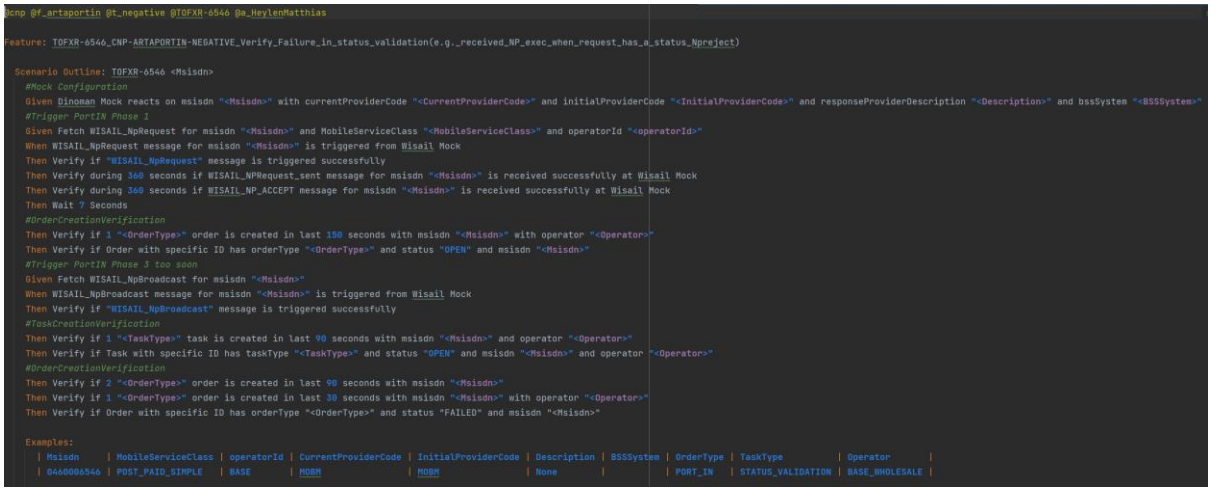
A Port-In test from Base who to Base Retail.



```
@cnp @f_portin @t_positive @TOFXR-6437 @a_HeylenMatthias

Feature: TOFXR-6437_CNP-PORTIN-POSITIVE_Verify_Port_In_from_Base_Wholesale_to_Base_Retail

  Scenario Outline: TOFXR-6437 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortOUT Phase 1
    Given Fetch NC_MobilePortInRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and RecipientServiceProviderCode "<RecipientServiceProviderCode>"
    When NC_MobilePortInRequest message for msisdn "<Msisdn>" is triggered from NC BSS Mock to CNP
    Then Verify if "NC_MobilePortInRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_REQUEST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTIN_REQUEST_SENT message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    #Trigger PortOUT Phase 2
    Given Fetch WISAIL_NpAccept for msisdn "<Msisdn>"
    When WISAIL_NpAccept message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpAccept" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortInAccept message for msisdn "<Msisdn>" is received successfully at NC BSS Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortOUT Phase 3
    Given Fetch NC_MobilePortInExec for msisdn "<Msisdn>"
    When NC_MobilePortInExec message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobilePortInExec" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NP_EXEC message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if NC_PORTIN_EXEC_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    #Trigger PortOUT Phase 4
    Given Fetch WISAIL_NpReady for msisdn "<Msisdn>"
    When WISAIL_NpReady message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpReady" message is triggered successfully
    Then Verify during 360 seconds if NC_MobilePortInReady message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Wait 7 Seconds
    #Trigger PortOUT Phase 5
    Given Fetch NC_MobilePortInRFS for msisdn "<Msisdn>"
    When NC_MobilePortInRFS message for msisdn "<Msisdn>" is triggered from NC OSS Mock to CNP
    Then Verify if "NC_MobilePortInRFS" message is triggered successfully
    Then Verify during 360 seconds if NC_PORTIN_RFS_SENT message for msisdn "<Msisdn>" is received successfully at NC OSS Mock
    Then Verify during 360 seconds if WISAIL_NP_BROADCAST message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | MobileServiceClass | RecipientServiceProviderCode | CurrentProviderCode | InitialProviderCode | OrderType | Description | BSSSystem   | Operator     |
      | 0460006437 | PRE_PAID           | BASE                         | BASE                | BASE                | PORT_IN   | None        | BASE LEGACY | BASE_RETAIL  |
```

CNP-ARTAPORTIN-NEGATIVE Verify Failure in status validation(e.g. received NP exec when request has a status Npreject)

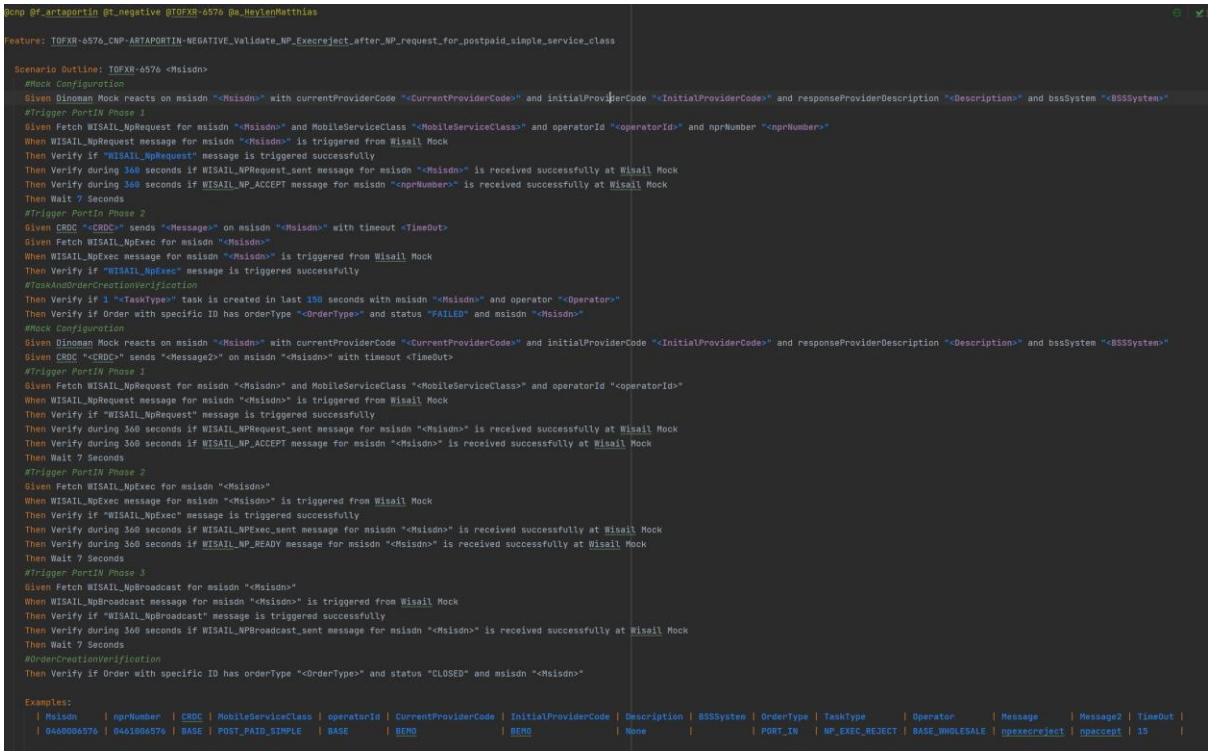
In this negative test we skip Phase 2 in the Port-In process to see if a Status Validation task is being created correctly.



```
@cnp @f_artaportin @t_negative @TOFXR-6546 @a_HeylenMatthias

Feature: TOFXR-6546_CNP-ARTAPORTIN-NEGATIVE_Verify_Failure_in_status_validation(e.g._received_NP_exec_when_request_has_a_status_Npreject)

  Scenario Outline: TOFXR-6546 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortIN Phase 1
    Given Fetch WISAIL_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>"
    When WISAIL_NpRequest message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPRequest_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if WISAIL_NP_ACCEPT message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if 1 "<OrderType>" order is created in last 150 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "OPEN" and msisdn "<Msisdn>"
    #Trigger PortIN Phase 3 too soon
    Given Fetch WISAIL_NpBroadcast for msisdn "<Msisdn>"
    When WISAIL_NpBroadcast message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpBroadcast" message is triggered successfully
    #TaskCreationVerification
    Then Verify if 1 "<TaskType>" task is created in last 90 seconds with msisdn "<Msisdn>" and operator "<Operator>"
    Then Verify if Task with specific ID has taskType "<TaskType>" and status "OPEN" and msisdn "<Msisdn>" and operator "<Operator>"
    #OrderCreationVerification
    Then Verify if 2 "<OrderType>" order is created in last 90 seconds with msisdn "<Msisdn>"
    Then Verify if 1 "<OrderType>" order is created in last 30 seconds with msisdn "<Msisdn>" with operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "FAILED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | MobileServiceClass | operatorId | CurrentProviderCode | InitialProviderCode | Description | BSSSystem | OrderType | TaskType        | Operator      |
      | 0460006546 | POST_PAID_SIMPLE   | BASE       | MOBM                | MOBM                | None        |           | PORT_IN   | STATUS_VALIDATION | BASE_WHOLESALE |
```

CNP-ARTAPORTIN-NEGATIVE Validate NP Execreject after NP request for postpaid simple service class

This scenario is another test which is supposed to fail after an Exec Reject message. After validating the task creation we check if the Port-In for the same msisdn can still be successful.

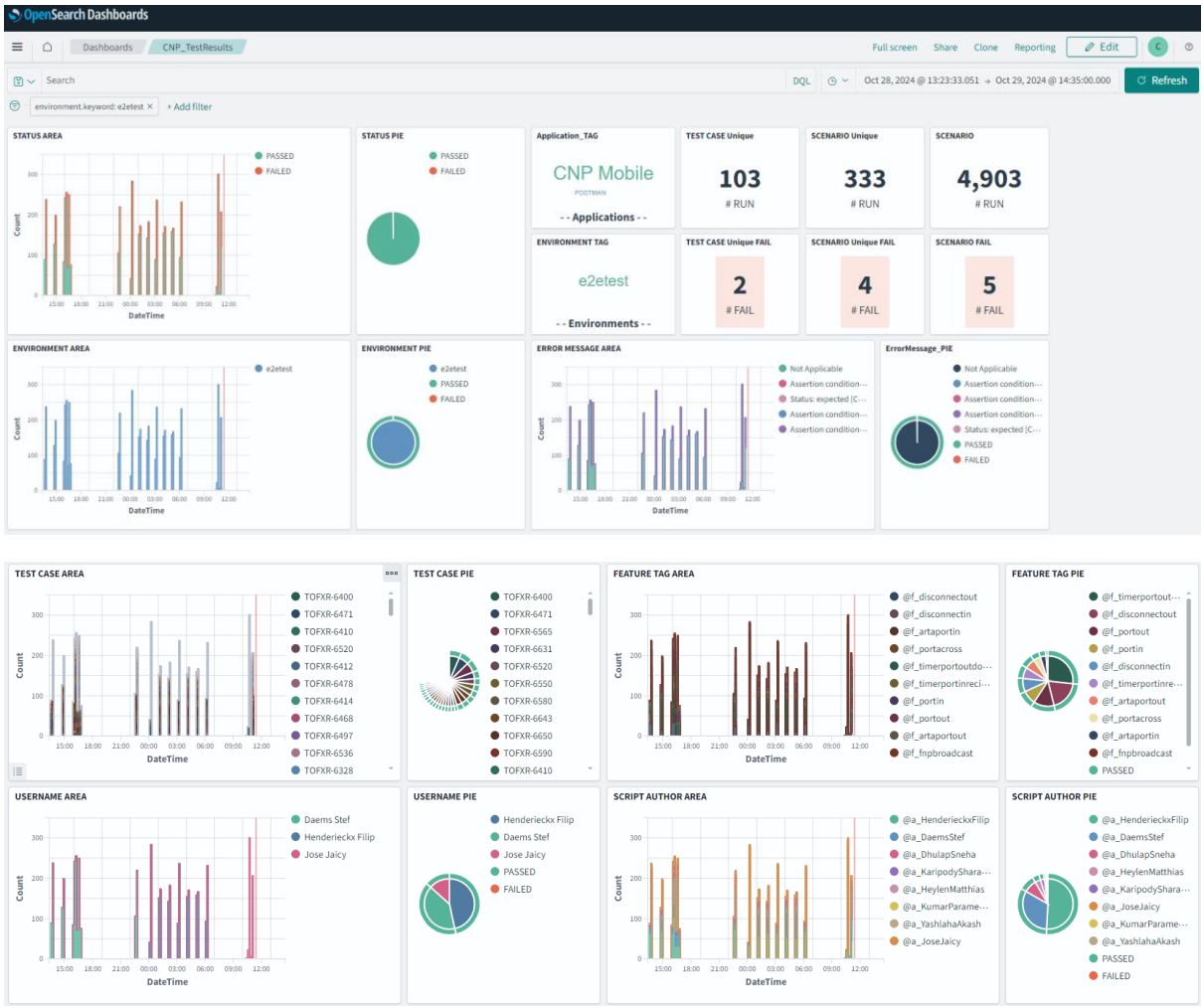


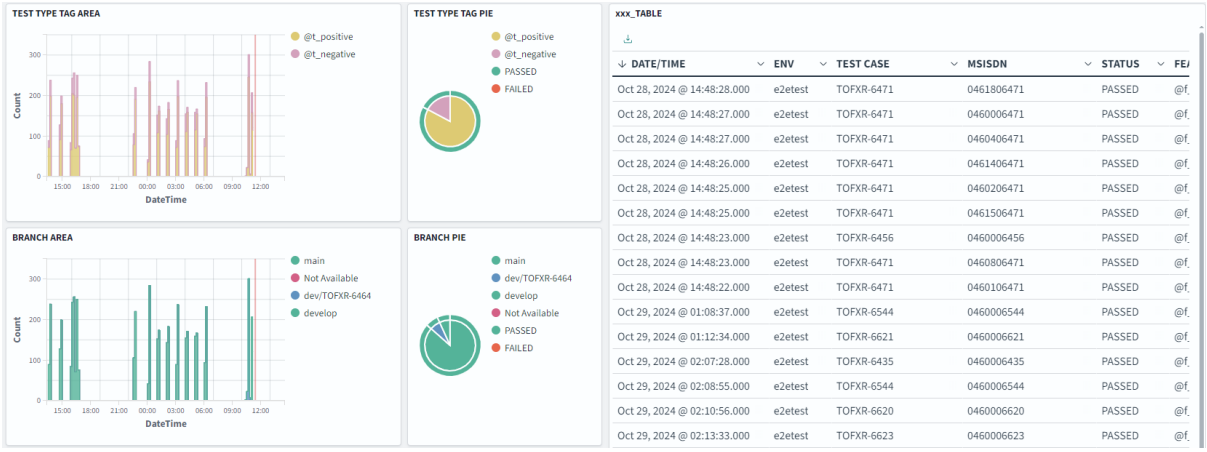
```
@cnp @f_artaportin @t_negative @TOFXR-6576 @a_HeylenMatthias

Feature: TOFXR-6576_CNP-ARTAPORTIN-NEGATIVE_Validate_NP_Execreject_after_NP_request_for_postpaid_simple_service_class

  Scenario Outline: TOFXR-6576 <Msisdn>
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    #Trigger PortIN Phase 1
    Given Fetch WISAIL_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>" and nprNumber "<nprNumber>"
    When WISAIL_NpRequest message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPRequest_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if WISAIL_NP_ACCEPT message for msisdn "<nprNumber>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #Trigger PortIn Phase 2
    Given CRDC "<CRDC>" sends "<Message>" on msisdn "<Msisdn>" with timeout <TimeOut>
    Given Fetch WISAIL_NpExec for msisdn "<Msisdn>"
    When WISAIL_NpExec message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpExec" message is triggered successfully
    #TaskAndOrderCreationVerification
    Then Verify if 1 "<TaskType>" task is created in last 150 seconds with msisdn "<Msisdn>" and operator "<Operator>"
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "FAILED" and msisdn "<Msisdn>"
    #Mock Configuration
    Given Dinoman Mock reacts on msisdn "<Msisdn>" with currentProviderCode "<CurrentProviderCode>" and initialProviderCode "<InitialProviderCode>" and responseProviderDescription "<Description>" and bssSystem "<BSSSystem>"
    Given CRDC "<CRDC>" sends "<Message2>" on msisdn "<Msisdn>" with timeout <TimeOut>
    #Trigger PortIN Phase 1
    Given Fetch WISAIL_NpRequest for msisdn "<Msisdn>" and MobileServiceClass "<MobileServiceClass>" and operatorId "<operatorId>"
    When WISAIL_NpRequest message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpRequest" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPRequest_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if WISAIL_NP_ACCEPT message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #Trigger PortIN Phase 2
    Given Fetch WISAIL_NpExec for msisdn "<Msisdn>"
    When WISAIL_NpExec message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpExec" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPExec_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Verify during 360 seconds if WISAIL_NP_READY message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #Trigger PortIN Phase 3
    Given Fetch WISAIL_NpBroadcast for msisdn "<Msisdn>"
    When WISAIL_NpBroadcast message for msisdn "<Msisdn>" is triggered from Wisail Mock
    Then Verify if "WISAIL_NpBroadcast" message is triggered successfully
    Then Verify during 360 seconds if WISAIL_NPBroadcast_sent message for msisdn "<Msisdn>" is received successfully at Wisail Mock
    Then Wait 7 Seconds
    #OrderCreationVerification
    Then Verify if Order with specific ID has orderType "<OrderType>" and status "CLOSED" and msisdn "<Msisdn>"

    Examples:
      | Msisdn     | nprNumber  | CRDC | MobileServiceClass | operatorId | CurrentProviderCode | InitialProviderCode | Description | BSSSystem | OrderType | TaskType       | Operator       | Message      | Message2 | TimeOut |
      | 0460006576 | 0461006576 | BASE | POST_PAID_SIMPLE   | BASE       | DEMO                | DEMO                | None        |           | PORT_IN   | NP_EXEC_REJECT | BASE_WHOLESALE | npexecreject | npaccept | 15      |
```

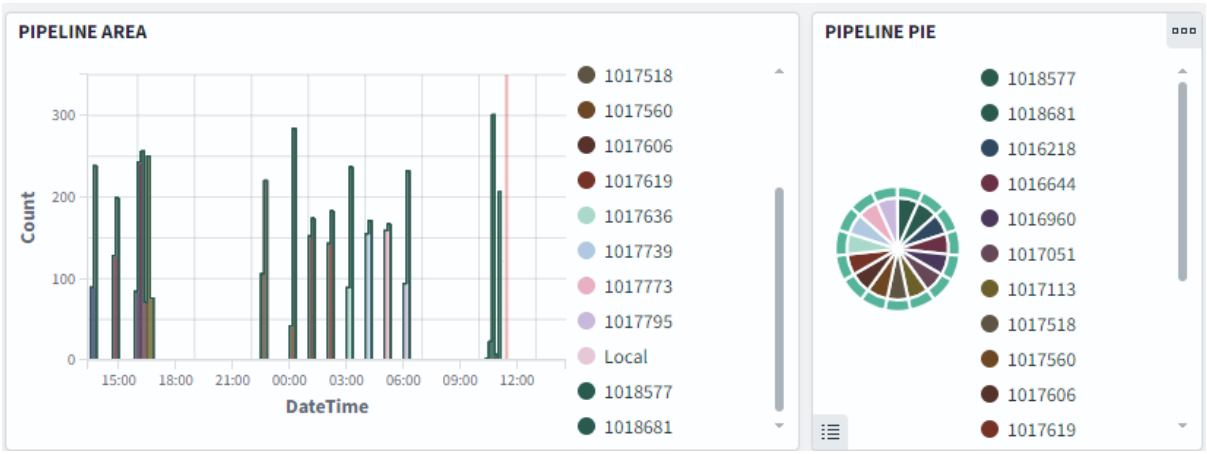
Dashboard e2e testing

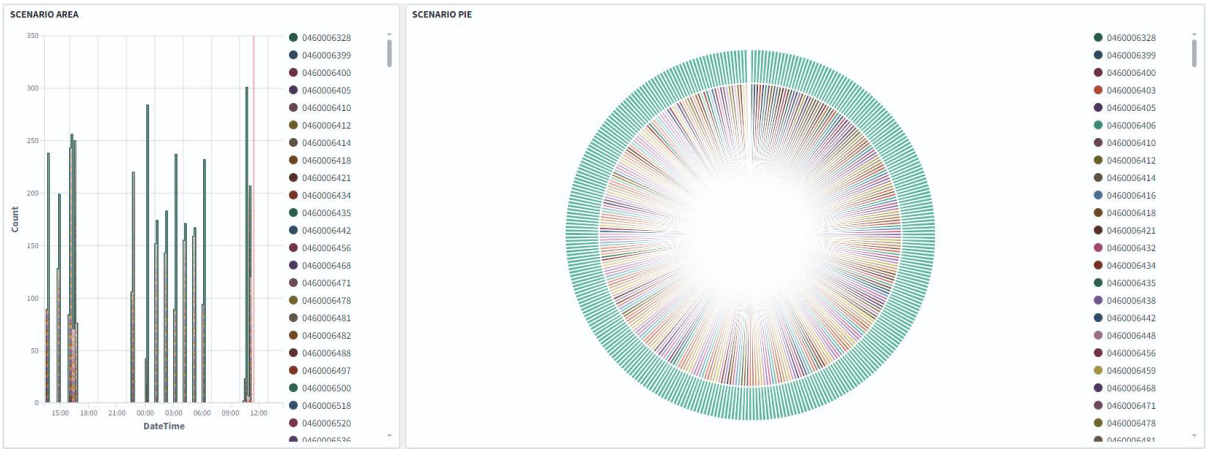
This Kibana dashboard is used to see the data of all automation tests. We can check which tests run successful or fail. Monitoring the dashboard constantly is important, because we can use it to see whether new changes done by the development team are passing our tests or not.

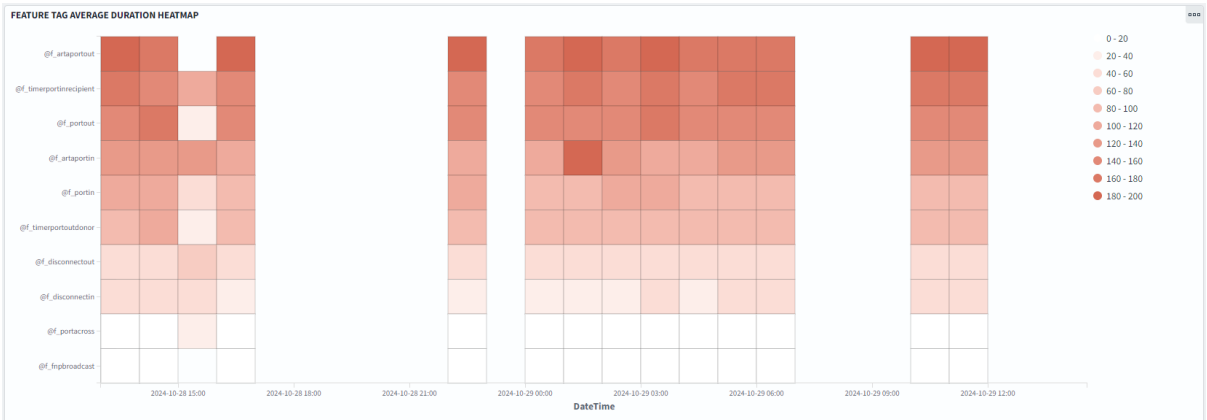
The data the dashboard provides makes it much easier to hunt down new bugs or other new problems in the code, which is where the value of the automation tests shines.

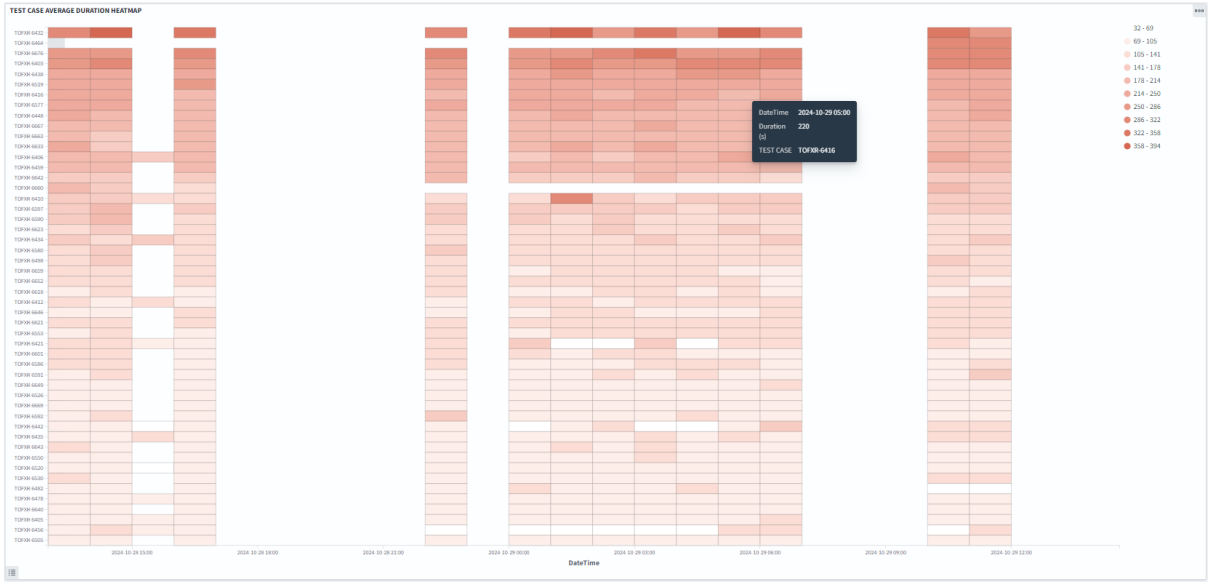



TEST TYPE TAG AREA
@t_positive
@t_negative
Count
DateTime
TEST TYPE TAG PIE
@t_positive
@t_negative
PASSED
FAILED
BRANCH AREA
main
Not Available
dev/TOFXR-6464
develop
Count
DateTime
BRANCH PIE
main
dev/TOFXR-6464
develop
Not Available
PASSED
FAILED
xxx_TABLE
DATE/TIME
ENV
TEST CASE
MSISDN
STATUS
Oct 28, 2024 @ 14:48:28.000 e2etest TOFXR-6471 0461806471 PASSED
Oct 28, 2024 @ 14:48:27.000 e2etest TOFXR-6471 0460006471 PASSED
Oct 28, 2024 @ 14:48:27.000 e2etest TOFXR-6471 0460406471 PASSED
Oct 28, 2024 @ 14:48:26.000 e2etest TOFXR-6471 0461406471 PASSED
Oct 28, 2024 @ 14:48:25.000 e2etest TOFXR-6471 0460206471 PASSED
Oct 28, 2024 @ 14:48:25.000 e2etest TOFXR-6471 0461506471 PASSED
Oct 28, 2024 @ 14:48:23.000 e2etest TOFXR-6456 0460006456 PASSED
Oct 28, 2024 @ 14:48:23.000 e2etest TOFXR-6471 0460806471 PASSED
Oct 28, 2024 @ 14:48:22.000 e2etest TOFXR-6471 0460106471 PASSED
Oct 29, 2024 @ 01:08:37.000 e2etest TOFXR-6544 0460006544 PASSED
Oct 29, 2024 @ 01:12:34.000 e2etest TOFXR-6621 0460006621 PASSED
Oct 29, 2024 @ 02:07:28.000 e2etest TOFXR-6435 0460006435 PASSED
Oct 29, 2024 @ 02:08:55.000 e2etest TOFXR-6544 0460006544 PASSED
Oct 29, 2024 @ 02:10:56.000 e2etest TOFXR-6620 0460006620 PASSED
Oct 29, 2024 @ 02:13:33.000 e2etest TOFXR-6623 0460006623 PASSED


PIPELINE AREA
Count
DateTime
1017518
1017560
1017606
1017619
1017636
1017739
1017773
1017795
Local
1018577
1018681
PIPELINE PIE
1018577
1018681
1016218
1016644
1016960
1017051
1017113
1017518
1017560
1017606
1017619


SCENARIO AREA
Count
DateTime
SCENARIO PIE


FEATURE TAG AVERAGE DURATION HEATMAP
@f_artaportout
@f_timerportinrecipient
@f_portout
@f_artaportin
@f_portin
@f_timerportoutdonor
@f_disconnectout
@f_disconnectin
@f_portacross
@f_fnpbroadcast
DateTime
0 - 20
20 - 40
40 - 60
60 - 80
80 - 100
100 - 120
120 - 140
140 - 160
160 - 180
180 - 200



DATE/TIME	ENV	TEST CASE	MSISDN	STATUS	FEATURE	AUTHOR	ERROR MESSAGE	EXECUTION DURATION	Count
Oct 29, 2024 @ 11:14:57.000	e2etest	TOFXR-6667	0460206667	PASSED	@f_portout	@a_HenderieckxFilip	Not Applicable	203	1
Oct 29, 2024 @ 11:15:00.000	e2etest	TOFXR-6667	0460506667	PASSED	@f_portout	@a_HenderieckxFilip	Not Applicable	196	1
Oct 29, 2024 @ 11:15:01.000	e2etest	TOFXR-6667	0460406667	PASSED	@f_portout	@a_HenderieckxFilip	Not Applicable	210	1
Oct 29, 2024 @ 11:15:03.000	e2etest	TOFXR-6667	0460306667	PASSED	@f_portout	@a_HenderieckxFilip	Not Applicable	210	1
Oct 29, 2024 @ 11:16:43.000	e2etest	TOFXR-6676	0460006676	PASSED	@f_timerportinrecipient	@a_HenderieckxFilip	Not Applicable	302	1
Oct 28, 2024 @ 16:37:01.000	e2etest	TOFXR-6512	0460006512	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	79	1
Oct 28, 2024 @ 16:01:49.000	e2etest	TOFXR-66O5	0460006605	PASSED	@f_portout	@a_HeylenMatthias	Not Applicable	40	1
Oct 28, 2024 @ 16:19:24.000	e2etest	TOFXR-6621	0460006621	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	107	1
Oct 29, 2024 @ 10:42:15.000	e2etest	TOFXR-6512	0460006512	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	44	1
Oct 28, 2024 @ 13:40:49.000	e2etest	TOFXR-6596	0460106596	PASSED	@f_fnpbroadcast	@a_HeylenMatthias	Not Applicable	0	1
Oct 29, 2024 @ 02:10:00.000	e2etest	TOFXR-6596	0460006596	PASSED	@f_fnpbroadcast	@a_HeylenMatthias	Not Applicable	0	1
Oct 29, 2024 @ 04:10:53.000	e2etest	TOFXR-6620	0460006620	PASSED	@f_portacross	@a_HeylenMatthias	Not Applicable	3	1
Oct 29, 2024 @ 10:42:10.000	e2etest	TOFXR-6544	0460006544	PASSED	@f_portacross	@a_HeylenMatthias	Not Applicable	3	1
Oct 29, 2024 @ 11:07:29.000	e2etest	TOFXR-6512	0460006512	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	44	1
Oct 28, 2024 @ 13:43:25.000	e2etest	TOFXR-6621	0460006621	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	114	1
									4,903
Oct 29, 2024 @ 11:07:26.000	e2etest	TOFXR-6544	0460006544	PASSED	@f_portacross	@a_HeylenMatthias	Not Applicable	3	1
Oct 28, 2024 @ 13:37:57.000	e2etest	TOFXR-6435	0460006435	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	79	1
Oct 28, 2024 @ 13:41:01.000	e2etest	TOFXR-6526	0460006526	PASSED	@f_artaportout	@a_HeylenMatthias	Not Applicable	99	1
Oct 28, 2024 @ 13:41:57.000	e2etest	TOFXR-66O5	0460006605	PASSED	@f_portout	@a_HeylenMatthias	Not Applicable	60	1
Oct 28, 2024 @ 14:50:13.000	e2etest	TOFXR-6596	0460006596	PASSED	@f_fnpbroadcast	@a_HeylenMatthias	Not Applicable	0	1
Oct 28, 2024 @ 14:51:21.000	e2etest	TOFXR-6620	0460006620	PASSED	@f_portacross	@a_HeylenMatthias	Not Applicable	3	1
Oct 28, 2024 @ 14:52:58.000	e2etest	TOFXR-6621	0460006621	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	107	1
Oct 28, 2024 @ 14:54:22.000	e2etest	TOFXR-6649	0460006649	PASSED	@f_artaportout	@a_HeylenMatthias	Not Applicable	99	1
Oct 28, 2024 @ 15:59:58.000	e2etest	TOFXR-6512	0460006512	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	44	1
Oct 28, 2024 @ 16:04:50.000	e2etest	TOFXR-6649	0460006649	PASSED	@f_artaportout	@a_HeylenMatthias	Not Applicable	99	1
Oct 28, 2024 @ 16:16:07.000	e2etest	TOFXR-6512	0460006512	PASSED	@f_artaportin	@a_HeylenMatthias	Not Applicable	37	1
Oct 28, 2024 @ 16:18:09.000	e2etest	TOFXR-66O5	0460006605	PASSED	@f_portout	@a_HeylenMatthias	Not Applicable	40	1
Oct 28, 2024 @ 16:37:16.000	e2etest	TOFXR-6596	0460006596	PASSED	@f_fnpbroadcast	@a_HeylenMatthias	Not Applicable	0	1
Oct 28, 2024 @ 16:37:33.000	e2etest	TOFXR-6526	0460006526	PASSED	@f_artaportout	@a_HeylenMatthias	Not Applicable	99	1
									4,903

Summary

Automation testing has proven to be a vital component of software development, and my internship at Telenet provided me with the opportunity to engage deeply with this aspect of quality assurance. Throughout my time with the Number Porting Tribe, I leveraged tools like Cucumber to design and execute automated test scenarios, ensuring the reliability and robustness of the fixed and mobile number porting systems. My work was instrumental in maintaining system integrity during development cycles, as the automation reduced the reliance on time-intensive manual testing and provided consistent results.

One of my key contributions was writing new test cases tailored to the specific needs of the project. By doing so, I helped ensure that the automated tests reflected the functional requirements of the system. Additionally, when unique scenarios demanded it, I defined new Cucumber steps, enabling the tests to address more nuanced or complex behaviors. This process required a deep understanding of the system's requirements and thorough collaboration with team members to align testing strategies with development goals.

The automation efforts not only expedited the testing phase but also enhanced the project's overall efficiency by detecting potential issues earlier in the development lifecycle. This proactive approach contributed to delivering a more stable product and reduced the risk of errors reaching production.

Overall, my internship at Telenet offered a comprehensive experience in automation testing. It underscored the importance of such practices in ensuring the reliability of software systems and demonstrated how strategic test automation can support both technical and organizational objectives. My contributions helped streamline the testing pipeline and reinforced the delivery of high-quality software solutions, leaving me with valuable skills and experiences that will serve me well in my future career.