**Test plan [Session Manager] POC**

1. **Session**
   1. **Session Creation**
2. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Create session** | | | |
| **Summary:**  The aim of this test is to verify session creation steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize Post attributes:  user\_id  duration  authn\_date  authn\_method  notification\_duration | 1. Verify the post response   201 created  {  session\_id  user\_id  expiration\_date  authn\_date  authn\_method  }   1. Verify that   expiration\_date=duration+ ActualDate |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Session creation steps** | | | |
| **Summary:**  The aim of this test is to verify session creation steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize Post attributes:  user\_id  duration  authn\_date  authn\_method  notification\_duration | 1. Session manager verify that the consumer role is WebAgent 2. Session Manager generates session\_id 3. Session Manager calculates ‘expiration\_date’ 4. Session manager add the new session on session table 5. Session manager calculates expireAt=timestamp(notification\_duration) 6. Add notification object to notification table 7. Verify notification attributes:   {  consumer\_id  session\_id  expireAt  notification\_expiration\_date  } |
| **Execution type**: | Automatic ok | | |

1. **Negative Scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized Consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer creates session on session manager:  Post /sessions | Initialize Post attributes:  user\_id  duration  authn\_date  authn\_method  notification\_duration | 401 Unauthorized |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Session creation by Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify that the Admin consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates session on session manager:  Post /sessions | Initialize Post attributes:  user\_id  duration  authn\_date  authn\_method  notification\_duration | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

* **Duration parameter**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Undefined duration** | | | |
| **Summary:**  The aim of this test is to verify that the session never expire when the duration is not initialized | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Do not initialize duration attribute | 1. Verify the post response   201 created  {  session\_id  user\_id  expiration\_date  authn\_date  authn\_method  }   1. Verify that   expiration\_date=Undefined |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of duration** | | | |
| **Summary:**  The aim of this test is to verify the response of the post when the duration format is bad. | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize duration attribute with bad format | **400 Bad request?** |
| **Execution type**: | Automatic ko | | |

* **user\_id parameter**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty user\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of the post when user\_id is not initialized | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Do not initialize user\_id attribute | **400 Bad request ?** |
| **Execution type**: | Automatic ko | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of user\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of the post with bad format of user\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize user\_id attribute with bad format | **400 Bad request ?** |
| **Execution type**: | Automatic ko | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Not existed user\_id !!!!!!!!!!!!!!!!!!!!!** | | | |
| **Summary:**  The aim of this test is to verify the response of the post with not existed user\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Do not initialize user\_id attribute | **400 Bad request ?** |
| **Execution type**: | Automatic ko | | |

* **authn\_method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty authn\_method** | | | |
| **Summary:**  The aim of this test is to verify the response when authn\_method is empty | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Do not initialize  authn\_method | Verify the post response  201 created  **{**  **session\_id**  **user\_id**  **expiration\_date**  **authn\_date**  **}** |
| **Execution type**: | Automatic ko | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of authn\_method** | | | |
| **Summary:**  The aim of this test is to verify the response of post with bad format of authn\_method | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize  authn\_method with bad format | **400 Bad request?** |
| **Execution type**: | Automatic ko | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Not existed authn\_method** | | | |
| **Summary:**  The aim of this test is to verify the response of post with bad format of authn\_method | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize  authn\_method with not existed value | **400 Bad request?** |
| **Execution type**: | Automatic ko | | |

* **Authn\_date**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty authn\_date** | | | |
| **Summary:**  The aim of this test is to verify the response of post with empty authn\_date | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Do not initialize  authn\_date | Verify the post response  201 created  **{**  **session\_id**  **user\_id**  **expiration\_date**  **authn\_method**  **}** |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of authn\_date** | | | |
| **Summary:**  The aim of this test is to verify the response of post with bad format of authn\_date | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize  authn\_date with bad format | **400 Bad request** |
| **Execution type**: | Automatic ko | | |

* **notification\_duration**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty notification\_duration** | | | |
| **Summary:**  The aim of this test is to verify the post response when notification\_duration is empty | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Do not initialize  notification\_duration | Verify the post response  201 created  **{**  **session\_id**  **user\_id**  **expiration\_date**  **authn\_method**  **}** |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of notification\_duration** | | | |
| **Summary:**  The aim of this test is to verify the post response when notification\_duration with bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize  notification\_duration  with bad format | **400 Bad request** |
| **Execution type**: | Automatic ko | | |

* 1. **Get Session** 
     1. **Get Session by session\_id**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get session** | | | |
| **Summary:**  The aim of this test is to verify get session by session\_id response | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by session\_id  Get /sessions/{session\_id} |  | Verify the get response  200 OK  {  session\_id  user\_id  expiration\_date  authn\_date  authn\_method  } |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by session\_id steps** | | | |
| **Summary:**  The aim of this test is to verify get session by session\_id steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by session\_id  Get /sessions/{session\_id} |  | 1. Session manager verify that the consumer role is WebAgent 2. Session manager sends the request to DB 3. DB returns the session 4. Session manager forwards the response to WebAgent |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Post and Get session** | | | |
| **Summary:**  The aim of this test is to verify that Post | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates session on session manager:  Post /sessions | Initialize those attributes:  user\_id  duration  authn\_date  authn\_method  notification\_duration | Verify the post response  201 created  {  session\_id  user\_id  expiration\_date  authn\_date  authn\_method  } |
| 2 | Get /sessions/{session\_id} |  | Verify the get response  200 OK  {  session\_id  user\_id  expiration\_date  authn\_date  authn\_method  } |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer get session by session\_id  Get /sessions/{session\_id} |  | 401 Unauthorized |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify that the Admin consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get session by session\_id  Get /sessions/{session\_id} |  | **403 Unauthorized?** |
| **Execution type**: | Automatic ok | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by not existed session\_id** | | | |
| **Summary:**  The aim of this test is to verify get session reponse by not existed session\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by not existed session\_id  Get /sessions/{user\_id} |  | **200 OK**  **{}** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by session\_id with bad format** | | | |
| **Summary:**  The aim of this test is to verify get session reponse by session\_id with bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by session\_id with bad format  Get /sessions/{session\_id} |  | **400 Bad request** |
| **Execution type**: | Automatic | | |

* + 1. **Get Session by user\_id**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get session** | | | |
| **Summary:**  The aim of this test is to verify that get sessions by user\_id returns all sessions associated to this user\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by user\_id  Get /sessions/{user\_id} |  | Verify the get response  200 OK  [{  Session\_id  User\_id  Expiration\_date  Authn\_date  Authn\_method  } ,{} ,{}] |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by user\_id steps** | | | |
| **Summary:**  The aim of this test is to verify get session by user\_id steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by user\_id  Get /sessions/{user\_id} |  | 1. Session manager verify that the consumer role is WebAgent 2. Session manager sends the request to DB 3. DB returns all sessions associated to this user\_id 4. Session manager forwards the response to WebAgent |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer get session by user\_id  Get /sessions/{ user\_id} |  | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify that the Admin consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get session by user\_id  Get /sessions/{ user\_id} |  | **4s01 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by not existed user\_id** | | | |
| **Summary:**  The aim of this test is to verify get session response by not existed user\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by not existed user\_id  Get /sessions/{user\_id} |  | **200 OK**  **[{}]** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by user\_id with bad format** | | | |
| **Summary:**  The aim of this test is to verify get session response by user\_id with bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session by user\_id with bad format  Get /sessions/{user\_id} |  | **400 Bad request** |
| **Execution type**: | Automatic | | |

* + 1. **Get Session All**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get session** | | | |
| **Summary:**  The aim of this test is to verify that get sessions returns all sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get session  Get /sessions |  | Verify the get response  200 OK  [{  session\_id  user\_id  expiration\_date  authn\_date  authn\_method  } ,{} ,{}] |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all session steps** | | | |
| **Summary:**  The aim of this test is to verify get all sessions steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get sessions  Get /sessions |  | 1. Session manager verify that the consumer role is WebAgent 2. Session manager sends the request to DB 3. DB returns all valid sessions 4. Session manager forwards the response to WebAgent |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer get all sessions  Get /sessions/ |  | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Session by Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify that the Admin consumer is unauthorized to manage sessions | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get all sessions  Get /sessions/ |  | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

* 1. **Delete Session**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Delete session** | | | |
| **Summary:**  The aim of this test is to verify delete session response | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent deletes session  Delete /sessions/{session\_id} |  | 204 No content |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Delete Session steps** | | | |
| **Summary:**  The aim of this test is to verify get all sessions steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent deletes session  Delete /sessions/{session\_id} |  | 1. Session manager verify that the consumer role is WebAgent 2. Session manager verify that the session\_id is existing 3. Session manager delete the table on DB 4. Session manager notify consumers that are subscribed to this session 5. Session manager delete notification tables associated to this session |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Delete session by Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify that admin consumer is unauthorized to delete session | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin deletes session by session\_id  Delete /sessions/{session\_id} |  | **401 unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to delete session | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer deletes session by session\_id  Delete /sessions/{session\_id} |  | 401 Unauthorized |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty session\_id** | | | |
| **Summary:**  The aim of this test is to verify session creation steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Delete /sessions |  | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : delete not existed Session** | | | |
| **Summary:**  The aim of this test is to verify delete session response by not existed session\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Delete /sessions/{session\_id} |  | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : delete Session with bad format of session\_id** | | | |
| **Summary:**  The aim of this test is to verify delete session response by session\_id with bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Delete /sessions/{session\_id} |  | **400 Bad request?** |
| **Execution type**: | Automatic | | |

* 1. **Subscription to a session**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Subscribe to a session response** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session response | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent subscribes to a session on session manager:  Post /sessions/{session\_id}/Watch | Initialize:  Notification\_duration | Verify the post response:  201 created |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Subscribe to a session steps** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent subscribes to a session on session manager:  Post /sessions/{session\_id}/Watch | Initialize notification\_duration | 1. Session manager verify that the consumer role is WebAgent 2. Session Manager verify that session\_id is existed 3. Session Manager calculates ‘expire\_At’ 4. Session manager add the new notification on notification table |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Subscribe to session by Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify that admin consumer is unauthorized to update session | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin consumer subscribe to session by session\_id  Post /sessions/{session\_id}/Watch | Initialize  notification\_duration | **401 unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to update session | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer updates session by session\_id  Post /sessions/{session\_id}/Watch | Initialize  notification\_duration | 401 Unauthorized |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty session\_id** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session response when session\_id is empty | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Post /sessions/Watch | Initialize:  notification\_duration | **400 Bad request** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Not existed Session** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session response when session\_id is not existed | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Post /sessions/{session\_id}/Watch | Initialize  notification\_duration | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of session\_id** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session response when session\_id is on bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Post /sessions/{session\_id}/Watch | Initialize  notification\_duration | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty notification\_duration** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session response when notification\_duration is empty | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Post /sessions/{}session\_id/Watch | Do not initialize:  notification\_duration | **?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of notification\_duration** | | | |
| **Summary:**  The aim of this test is to verify subscription to a session response when notification\_duration is on bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Post /sessions/{session\_id}/Watch | Initialize  notification\_duration  With bad format | **400 Bad request?** |
| **Execution type**: | Automatic | | |

1. **Consumer**
   1. **Admin**
      1. **Create Admin Consumer**
2. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Create Admin Consumer** | | | |
| **Summary:**  The aim of this test is to verify the response of Admin consumer creation | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates Admin consumer:  Post /admin/consumers | Initialize consumer attributes:  consumer\_login  consumer\_password | 1. Verify the post response   201 created  {  consumer\_id  consumer\_login  consumer\_password  consumer\_role  }   1. Verify   consumer\_role==Admin |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Admin consumer creation steps** | | | |
| **Summary:**  The aim of this test is to verify admin consumer creation steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates consumer  Post /admin/consumers | Initialize consumer attributes:  consumer\_login  consumer\_password | 1. Session manager verify that the consumer role is Admin 2. Session Manager generates consumer\_id 3. Session manager add the new Admin consumer on consumer table |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Admin creation by WebAgent** | | | |
| **Summary:**  The aim of this test is to verify that the WebAgent consumer is unauthorized to create admin consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates admin consumer  Post /admin/consumers | Initialize consumer attributes:  consumer\_login  consumer\_password | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unregistered consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to create admin consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer creates admin consumer  Post /admin/consumers | Initialize consumer attributes:  consumer\_login  consumer\_password | 401 Unauthorized |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Creation Admin consumer attributes** | | | |
| **Summary:**  The aim of this test is to verify that creation Admin consumer attributes are mandatory | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates admin consumer  Post /admin/consumers | Do not initialize:  consumer\_login | **401 Unauthorized?**  **400 Bad request?** |
| 1 | Admin creates admin consumer  Post /admin/consumers | Do not initialize:  consumer\_password | **401 Unauthorized?**  **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of Admin consumer attributes** | | | |
| **Summary:**  The aim of this test is to verify that creation Admin consumer response when attributes are on bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates admin consumer  Post /admin/consumers | Initialize:  consumer\_login with bad format  consumer\_password | **400 Bad request?** |
| 2 | Admin creates admin consumer  Post /admin/consumers | Initialize:  consumer\_login  consumer\_password with bad format | **400 Bad request?** |
| **Execution type**: | Automatic | | |

* + 1. **Get Consumer**
       1. **Get consumer by consumer\_id**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by consumer\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer by consumer\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get admin consumer by consumer\_id  Get /admin/consumers/{consumer\_id} |  | 200 OK  {  consumer\_id  consumer\_login  consumer\_password  consumer\_role  } |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by consumer\_id steps** | | | |
| **Summary:**  The aim of this test is to verify get consumer steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get admin consumer by consumer\_id  Get /admin/consumers/{consumer\_id} |  | 1. Session manager verify that the consumer role is Admin 2. Session manager sends the request to DB 3. DB returns the Admin consumer 4. Session manager forwards the response to Admin |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Post and Get consumer** | | | |
| **Summary:**  The aim of this test is to verify that get consumer is an inverse function of post consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates Admin consumer:  Post /admin/consumers | Initialize consumer attributes:  consumer\_login  consumer\_password | 1. Verify the post response   201 created  {  **consumer\_id**  **consumer\_login**  **consumer\_password**  **consumer\_role**  } |
| 2 | Get /admin/consumers/{consumer\_id} |  | 200 OK  {  **consumer\_id**  **consumer\_login**  **consumer\_password**  **consumer\_role**  } |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Admin by WebAgent** | | | |
| **Summary:**  The aim of this test is to verify that the WebAgent consumer is unauthorized to get Admin consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get admin consumer by consumer\_id  Get /admin/consumers/{consumer\_id} |  | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that the unauthorized consumer is unauthorized to get Admin consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unauthorized consumer get admin consumer by consumer\_id  Get /admin/consumers/{consumer\_id} |  | 401 Unauthorized |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by not existed consumer\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer request when the consumer\_id is not existed | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get admin consumer by not existed consumer\_id  Get /admin/consumers/{consumer\_id} |  | **200 OK**  **{}** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get Admin consumer by consumer\_id with bad format** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer request by the consumer\_id with bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get admin consumer by consumer\_id with bad format  Get /admin/consumers/{consumer\_id} |  | **400 Bad request** |
| **Execution type**: | Automatic | | |

* + - 1. **Get All consumer**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all Admin consumers** | | | |
| **Summary:**  The aim of this test is to verify that get consumers without parameters returns all Admin consumers | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get all admin consumer  Get /admin/consumers |  | 200 OK  [{  consumer\_id  consumer\_login  consumer\_password  consumer\_role  }, {}, {}  ] |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all Admin consumers steps** | | | |
| **Summary:**  The aim of this test is to verify get all admin consumers steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get all admin consumer  Get /admin/consumers |  | 1. Session manager verify that the consumer role is Admin 2. Session manager sends the request to DB 3. DB returns all admin consumers 4. Session manager forwards the response to the Admin |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all Admin consumers by WebAgent** | | | |
| **Summary:**  The aim of this test is to verify that WebAgent is unauthorized to get all Admin consumers | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get all admin consumer  Get /admin/consumers |  | **401 unauthorized** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to get all Admin consumers | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer get all admin consumer  Get /admin/consumers |  | **401 unauthorized** |
| **Execution type**: | Automatic | | |

* + 1. **Update Admin consumer**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Update Admin consumer** | | | |
| **Summary:**  The aim of this test is to verify update Admin consumer response | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin consumer 1 updates admin consumer 2 by consumer\_id  Patch /admin/consumers/{consumer\_id} | Reset consumer2 attributes:  consumer\_login  consumer\_password | 201 OK  {  consumer\_id  consumer\_login  consumer\_password  consumer\_role  } |
| 2 | Verify that the attributes of admin consumer2 is updated on DB |  |  |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Update Admin consumer steps** | | | |
| **Summary:**  The aim of this test is to verify update Admin consumer steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin 1 updates admin consumer2 by consumer\_id  Patch /admin/consumers/{consumer\_id} | Reset consumer2 attributes:  consumer\_login  consumer\_password | 1. Session manager verify that the consumer role is Admin 2. Session manager update the admin consumer on consumer table |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Update Admin consumer by WebAgent** | | | |
| **Summary:**  The aim of this test is to verify that WebAgent is unauthorized to update Admin consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent updates admin consumer with consumer\_id  Patch /admin/consumers/{consumer\_id} | Reset consumer2 attributes:  consumer\_login  consumer\_password | **401 Unauthorized** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized Consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to update Admin consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered Admin 1 updates admin consumer2 with consumer\_id  Patch /admin/consumers/{consumer\_id} | Reset consumer2 attributes:  consumer\_login  consumer\_password | **401 unauthorized** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Empty consumuer\_id** | | | |
| **Summary:**  The aim of this test is to verify update Admin consumer response when consumer\_id is empty | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin consumer 1 updates Admin consumer 2  Patch /admin/consumers | Reset consumer2 attributes:  consumer\_login  consumer\_password | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Not existed consumuer\_id** | | | |
| **Summary:**  The aim of this test is to verify update Admin consumer response when consumer\_id is not existed | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin 1 updates admin consumer2 with not existed consumer\_id  Patch /admin/consumers/{consumer\_id} | Reset consumer2 attributes:  consumer\_login  consumer\_password | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of consumuer\_id** | | | |
| **Summary:**  The aim of this test is to verify update Admin consumer response when consumer\_id is on bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin 1 updates admin consumer2 by consumer\_id with bad format  Patch /admin/consumers/{consumer\_id} | Reset consumer2 attributes:  consumer\_login  consumer\_password | **400 Bad request?** |
| **Execution type**: | Automatic | | |

* 1. **WebAgent**
     1. **Create WebAgent Consumer**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Create WebAgent Consumer** | | | |
| **Summary:**  The aim of this test is to verify the response of WebAgent consumer creation | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates WebAgent consumer:  Post /admin/WebAgents | Initialize WebAgent attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | 1. Verify the post response   201 created  {  consumer\_id  consumer\_login  consumer\_password  consumer\_role  notification\_uri  notification\_credentials  }   1. Verify   consumer\_role==WebAgent |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : WebAgent consumer creation steps** | | | |
| **Summary:**  The aim of this test is to verify WebAgent consumer creation steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates WebAgent consumer  Post /admin/WebAgents | Initialize WebAgent attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | 1. Session manager verify that the consumer role is Admin 2. Session Manager generates consumer\_id 3. Session manager add the new WebAgent consumer on consumer table |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : WebAgent creation by WebAgent** | | | |
| **Summary:**  The aim of this test is to verify that the WebAgent is unauthorized to create WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent creates WebAgent consumer  Post /admin/WebAgents | Initialize those parameters:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to create WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer creates WebAgent consumer  Post /admin/WebAgents | Initialize WebAgent attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | 401 Unauthorized |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Creation WebAgent consumer attributes** | | | |
| **Summary:**  The aim of this test is to verify that creation WebAgent consumer attributes are mandatory | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates WebAgent consumer  Post /admin/WebAgents | Do not initialize:  consumer\_login | **400 Bad request?** |
| 2 | Admin creates WebAgent consumer  Post /admin/WebAgents | Do not initialize:  consumer\_password | **400 Bad request?** |
| 3 | Admin creates WebAgent consumer  Post /admin/WebAgents | Do not initialize:  notification\_uri | **400 Bad request?** |
| 4 | Admin creates WebAgent consumer  Post /admin/WebAgents | Do not initialize:  notification\_credentials | **400 Bad request?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Bad format of WebAgent consumer attributes** | | | |
| **Summary:**  The aim of this test is to verify that the response of creation WebAgent consumer when attributes are on bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates WebAgent consumer  Post /admin/WebAgents | Initialize:  consumer\_login with bad format | **400 Bad request?** |
| 2 | Admin creates WebAgent consumer  Post /admin/WebAgents | Initialize:  consumer\_password with bad format | **400 Bad request?** |
| 3 | Admin creates WebAgent consumer  Post /admin/WebAgents | Initialize:  notification\_uri with bad format | **400 Bad request?** |
| 4 | Admin creates WebAgent consumer  Post /admin/WebAgents | Initialize:  notification\_credentials with bad format | **400 Bad request?** |
| **Execution type**: | Automatic | | |

* + 1. **Get Consumer**
       1. **Get consumer by consumer\_id**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by consumer\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of get WebAgent consumer by consumer\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get WebAgent consumer by consumer\_id  Get /admin/ WebAgents/{consumer\_id} |  | 200 OK  {  consumer\_id  consumer\_login  consumer\_password  consumer\_role  notification\_uri  notification\_credentials  } |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by consumer\_id steps** | | | |
| **Summary:**  The aim of this test is to verify get consumer steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get WebAgent consumer by consumer\_id  Get /admin/WebAgents/{consumer\_id} |  | 1. Session manager verify that the consumer role is Admin 2. Session manager sends the request to DB 3. DB returns the WebAgent consumer 4. Session manager forwards the response to Admin |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Post and Get consumer** | | | |
| **Summary:**  The aim of this test is to verify that get consumer is an inverse function of post consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin creates WebAgent consumer:  Post /admin/WebAgents | Initialize those parameters:  **consumer\_login**  **consumer\_password**  **notification\_uri**  **notification\_credentials** | 1. Verify the post response   201 created  {  **consumer\_id**  **consumer\_login**  **consumer\_password**  **consumer\_role**  **notification\_uri**  **notification\_credentials**  }   1. Verify   consumer\_role==WebAgent |
| 2 | Get /admin/WebAgents/{consumer\_id} |  | 200 OK  {  **consumer\_id**  **consumer\_login**  **consumer\_password**  **consumer\_role**  **notification\_uri**  **notification\_credentials**  } |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by WebAgent consumer** | | | |
| **Summary:**  The aim of this test is to verify that the WebAgent consumer is unauthorized to get WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent 1 get WebAgent consumer 2 by consumer\_id  Get /admin/WebAgents/{consumer\_id} |  | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to get WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer get WebAgent consumer 2 by consumer\_id  Get /admin/WebAgents/{consumer\_id} |  | 401 Unauthorized |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer with empty consumer\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer request when the consumer\_id is empty | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get WebAgent consumer by empty consumer\_id  Get /admin/WebAgents |  | **400 Bad request** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by not existed consumer\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer request when the consumer\_id is not existed | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get WebAgent consumer by not existed consumer\_id  Get /admin/WebAgents/{consumer\_id} |  | **200 OK**  **{}** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get consumer by Admin consumer\_id** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer by Admin consumer\_id | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get admin consumer by Admin consumer\_id  Get /admin/WebAgents/{consumer\_id} |  | **200 OK**  **{}** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get WebAgent consumer by consumer\_id with bad format** | | | |
| **Summary:**  The aim of this test is to verify the response of get consumer request by consumer\_id with bad format | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get admin consumer by consumer\_id with bad format  Get /admin/WebAgents/{consumer\_id} |  | **400 Bad request** |
| **Execution type**: | Automatic | | |

* + - 1. **Get All consumer**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all WebAgent consumers** | | | |
| **Summary:**  The aim of this test is to verify that get WebAgents consumers without parameters returns all WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get All WebAgent consumers  Get /admin/WebAgents |  | 200 OK  [{  consumer\_id  consumer\_login  consumer\_password  consumer\_role  notification\_uri  notification\_credentials  }, {}, {}  ] |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all WebAgent consumers steps** | | | |
| **Summary:**  The aim of this test is to verify get all WebAgent consumers steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin get All WebAgent consumers  Get /admin/WebAgents |  | 1. Session manager verify that the consumer role is Admin 2. Session manager sends the request to DB 3. DB returns all WebAgent consumers 4. Session manager forwards the response to the Admin consumer |
| **Execution type**: | Automatic | | |

1. **Negative scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Get all WebAgent consumers by WebAgent** | | | |
| **Summary:**  The aim of this test is to verify that get WebAgents consumers by WebAgent is not allowed | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent get All WebAgent consumers  Get /admin/WebAgents |  | **401 Unauthorized** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumers is unauthorized to get WebAgents | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer get All WebAgent consumers  Get /admin/WebAgents |  | 401 Unauthorized |
| **Execution type**: | Automatic | | |

* + 1. **Update WebAgent consumer**

1. **Nominal scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Update WebAgent consumer** | | | |
| **Summary:**  The aim of this test is to verify update WebAgent consumer response | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin updates WebAgent by consumer\_id  Patch /admin/WebAgents/{consumer\_id} | Reset WebAgent attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | 201  {  consumer\_id  consumer\_login  consumer\_password  consumer\_role  notification\_uri  notification\_credentials  } |
| 2 | Verify that the WebAgent consumer is updated on DB |  |  |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Update WebAgent consumer steps** | | | |
| **Summary:**  The aim of this test is to verify update WebAgent consumer steps | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Admin updates WebAgent by consumer\_id  Patch /admin/WebAgents/{consumer\_id} | Reset WebAgent attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | 1. Session manager verify that the consumer role is Admin 2. Session manager update the WebAgent consumer on consumer table |
| **Execution type**: | Automatic | | |

1. **Negative scenarios:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : WebAgent Update consumer** | | | |
| **Summary:**  The aim of this test is to verify that WebAgent is unauthorized to update WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | WebAgent 1 updates WebAgent 2 by consumer\_id  Patch /admin/WebAgents/{consumer\_id} | Reset WebAgent2 attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | **401 Unauthorized?** |
| **Execution type**: | Automatic | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case : Unauthorized consumer** | | | |
| **Summary:**  The aim of this test is to verify that unregistered consumer is unauthorized to update WebAgent consumer | | | |
|  | **Step actions:** | **Preconditions:** | **Expected Results:** |
| 1 | Unregistered consumer updates WebAgent 2 by consumer\_id  Patch /admin/WebAgents/{consumer\_id} | Reset WebAgent2 attributes:  consumer\_login  consumer\_password  notification\_uri  notification\_credentials | 401 Unauthorized |
| **Execution type**: | Automatic | | |