**Testing Plan**

**Purpose:**

Unit testing is an important step to ensure the code is reliable and robust. This document describes all use cause scenarios developers need to consider, to make each independent function works properly.

**Environment:**

This document focuses on the server side since there are only one util function on the client side. All unit tests described in this document were carried out using AWS Lambda Function tools.

**UNIT TEST (BACKEND)**

1. createTeleTAN():

* This method generates a random token with 6 digits
* Scenario 1:
  + Input: Invoke a get request
  + Response: Should return a 6 digits token
  + DB: A new record appears in table registration\_keys

1. getNewlyGeneratedPks(timestamp):

* This method returns all PKs uploaded after the input timestamp
* Scenario 1:
  + Input: Timestamp = 0
  + Response: Should return all PKs
* Scenario 2:
  + Input: Timestamp = t
  + Response: All returned PKs should have timestamp > t
* Scenario 3:
  + Input: Timestamp = current time
  + Response: No PK should be returned

1. getRegistrationKeyGUID(guid)
   * This method will return a registration key if the inputted GUID is valid
   * Scenario 1:
   * Input: GUID is empty
   * Response: Response status code should be 400
   * DB: Table registration\_keys not changed

* Scenario 2:
  + Input: GUID is not a string of 16 characters
  + Response: Response status code should be 400
  + DB: Table registration\_keys not changed
* Scenario 3:
  + Input: GUID is not in the database
  + Response: Response status code should be 400
  + DB: Table registration\_keys not changed
* Scenario 4:
  + Input: GUID is valid and is in the database
  + Response: Return with a registration key of length 32
  + DB: New record added for table registration\_keys
* Scenario 5:
  + Input: Use a valid GUID twice
  + Response: Return a new registration key of length 32
  + DB: In table registration\_keys, the value for field registration key hashed associated with this GUID gets updated. Number of records in table registration\_keys and tans remain the same

1. getRegistrationKeyTELETAN(TeleTAN)
   * This method will return a registration key if the inputted TELETAN is valid
   * Scenario 1:
   * Input: TELETAN is empty or is not a 6 digits String
   * Response: Response with “TELETAN INVALID”, and status code 400
   * DB: Table registration\_keys not changed

* Scenario 2:
  + Input: TELETAN is valid, but submitted after 30 minutes since generated
  + Response: Response with “TELETAN Timeout”, and status code 400
  + DB: Table registration\_keys not changed
* Scenario 3:
  + Input: TELETAN is valid, and submitted within 30 minutes since generated
  + Response: Return a new registration key of length 32
  + DB: Table registration\_keys gets updated
* Scenario 4:
  + Input: Submit a valid TELETAN twice within 30 minutes since generated
  + Response: Response with “TELETAN INVALID”, and status code 400
  + DB: Table registration\_keys not changed

1. getTAN(registration\_key)
   * This method returns a tan if the input is a valid registration\_key
   * Scenario 1:
   * Input: Using a random registration\_key as input
   * Response: Response with status code 400

* Scenario 2:
  + Input: Call getRegistrationKeyGUID first, use the return as input
  + Response: Should return a valid tan with status code 200
* Scenario 3:
  + Input: Call getRegistrationKeyTELETAN first, use the return as input
  + Response: Should return a valid tan with status code 200

1. Upload\_periodic\_key(periodic\_keys, api\_level, android\_version, brand, model, user\_id, tan)
   * This method check whether the input is valid, and then store them in the database
   * Scenario 1:
   * Input: TAN is valid, but periodic\_keys is an empty JSON
   * Response: Return “OK” with status code 200
   * DB: Nothing changed on table periodic\_keys

* Scenario 2:
  + Input: TAN is a random string, and is not valid
  + Response: Response with “Verify failed”, with status code being 400
* Scenario 3:
  + Input: TAN is valid, but gets reused
  + Response: Response with “Verify failed”, with status code being 400
* Scenario 4:
  + Input: TAN is valid, and passed verification, and periodic keys json is not empty
  + Response: Return “OK” with status code 200
  + DB: Records added on table periodic\_keys

1. VerifyTAN(tan)
   * This method checks if a tan is valid
   * Scenario 1:
   * Input: TAN a random string
   * Response: Return “FAILED”, with status code 400

* Scenario 2:
  + Input: TAN is valid, and is within 5 minutes since generated
  + Response: Return “SUCCESS”, with status code 200
* Scenario 3:
  + Input: TAN is valid, and is not within 5 minutes since generated
  + Response: Return “FAILED”, with status code 400