REST API tests overview

Integrations and Markets

Exported on 03/02/2023

Table of Contents

1 Purpose of document 8
2 Purpose of REST API automated tests 9
3 Prerequisites 10
4 Structure 11
5 Useful notes 12
6 Test Case steps description 13
7 Prerequisite test case 14
7.1 ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS 14
8 Test Cases 15
8.1 CHECK_USING_NEW_SID 15
8.2 DEBIT_BET_ALREADY_EXIST 15
8.3 DEBIT_CANCEL 15
8.4 DEBIT_CREDIT 16
8.5 DEBIT_CREDIT_BET_ALREADY_SETTLED 16
8.6 DEBIT_CREDIT_BET_DOES_NOT_EXIST 16
8.7 DEBIT_CREDIT_WITH_DIFFERENT_SIDS 17
8.8 SAME_USER_PLAYING_WITH_DIFFERENT_CHANNEL_TYPE 17
8.9 TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS 18
8.10 VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME 19
8.11 CHECK_USING_INVALID_USER_ID 19
8.12 Verifies that non-existing user ID can't be used to perform requests. 19
8.13 DEBIT_CANCEL_BET_DOES_NOT_EXIST 20
8.14 DEBIT_CANCEL_BET_ALREADY_SETTLED 21
8.15 CREDIT_WITH_ZERO_AMOUNT 21
8.16 DEBIT_CREDIT_SETTLE_TYPE_GAMEWISE 21
8.17 MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL 22
8.18 MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT 22
8.19 INSUFFICIENT_FUNDS 23

8.20 ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL 24
8.21 CRAPS_DELAYED_SETTLEMENT_Long_Living_Bet.....24
8.22 DEBIT_CREDIT_CANCEL.....25
8.23 For casinos that don't use depositAfterCancel 25
8.24 DEBIT_CANCEL_CREDIT.....25
8.25 DEBIT_CREDIT_WITH_BIG_PAYOUT 25
8.26 RESPONSE_TIME_VALIDATION.....26
8.27 TIP_DEBIT_CANCEL 26
8.28 TIP_DEBIT_CLOSE.....27
8.29 TIP_IDEMPOTENT_DEBIT 27
8.30 TIP_IDEMPOTENT_CANCEL 27
8.31 TIP_IDEMPOTENT_CLOSE 28
8.32 TIP_DEBIT_BETWEEN_GAMES 28
8.33 CANCEL_DEBIT_BEFORE_CANCEL_FOR_ANOTHER_GAME.....28
8.34 FREE_ROUND_PLAYABLE_SPENT_PROMO_PAYOUT.....29
8.35 DOUBLE_PROMO_PAYOUT.....29
8.36 JACKPOT_PROMO_PAYOUT 30
8.37 FROM_GAME_PROMO_PAYOUT 30
8.38 REAL_TIME_MONETARY_REWARD_PROMO_PAYOUT.....30
8.39 REWARD_GAME_MIN_BET_LIMIT_REACHED_PROMO_PAYOUT 31
8.40 REWARD_GAME_WIN_CAP_REACHED_PROMO_PAYOUT 31
8.41 REWARD_GAME_PLAYABLE_SPENT_PROMO_PAYOUT 31
8.42 NEW_TYPE_PROMO_PAYOUT.....32
8.43 DEBIT_CREDIT_WITH_EVO/NE/RT_JACKPOT_STRUCTURE.....32
8.44 DEBIT_CANCEL_WITH_EVO/NE/RT_JACKPOT_STRUCTURE.....32
8.45 PROMO_DEBIT_PROMO_CREDIT 33
8.46 PROMO_DEBIT_PROMO_CANCEL 33
8.47 DOUBLE_PROMO_DEBIT 33
8.48 DOUBLE_PROMO_CREDIT 34
8.49 DOUBLE_PROMO_CANCEL 34
8.50 PROMO_CREDIT_NO_DEBIT 34

8.51 PROMO_CANCEL_NO_DEBIT 35
8.52 MULTIPLE_DEBIT_CREDIT_CANCEL_SETTLE_TYPE_MIXED 35
8.53 DEBIT_CANCEL_BET_ALREADY_SETTLED_UNEXPECTED 35
8.54 DEBIT_CREDIT_BET_ALREADY_SETTLED_UNEXPECTED 36
9 AAMS Test Cases 37
9.1 AAMS_ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS 37
9.2 AAMS_SESSION_ALREADY_CLOSED 37
9.3 AAMS_SESSION_DOES_NOT_EXIST 37
9.4 AAMS_SESSION_ALREADY_CLOSED_CANCEL 38
9.5 AAMS_SESSION_ALREADY_CLOSED_CREDIT 38
9.6 AAMS_SESSION_ALREADY_CLOSED_DEBIT 38
9.7 AAMS_LOBBY_BALANCE 39
9.8 AAMS_DEBIT_CREDIT 39
9.9 AAMS_DEBIT_CANCEL 39
9.10 AAMS_DEBIT_CREDIT_CANCEL 40
9.11 AAMS_DEBIT_CANCEL_CREDIT 40
9.12 AAMS_DEBIT_BET_ALREADY_EXIST 40
9.13 AAMS_DEBIT_CREDIT_BET_DOES_NOT_EXIST 41
9.14 AAMS_DEBIT_CANCEL_BET_DOES_NOT_EXIST 41
9.15 AAMS_DEBIT_CREDIT_BET_ALREADY_SETTLED 42
9.16 AAMS_DEBIT_CANCEL_BET_ALREADY_SETTLED 42
9.17 AAMS_MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL 43
9.18 AAMS_MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT 43
9.19 AAMS_TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS 44
9.20 AAMS_ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL 45
9.21 AAMS_DEBIT_CREDIT_WITH_DIFFERENT_SIDS 45
9.22 AAMS_CHECK_USING_NEW_SID 46
9.23 AAMS_CREDIT_WITH_ZERO_AMOUNT 46
9.24 AAMS_DEBIT_CREDIT_SETTLE_TYPE_GAMEWISE 47
9.25 AAMS_VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME 47
10 REST API tests results 49

10.1 2. In horizontal menu tab select: onewallet-rest-api 49
10.2 3. In the vertical list select the desired casino by a click: 50
10.3 4. In the left window side select one of the builds by a click: 51
10.4 5. In the left menu bar select Console Output: 52
10.5 6. Scroll the page down to "TESTING COMPLETED" where you will find a summary of tests' execution: 52
11 SID for REST API tests 54
11.1 What are the ways to get SID value? Predefined SID value Non-predefined SID value What is SID method? Example of how it works with a non-predefined SID value: What are the ways to get SID value? 54
11.1.1 Predefined SID value 54
11.1.2 Non-predefined SID value 54
11.2 What is SID method? 54
11.3 Example of how it works with a non-predefined SID value: 54

- Purpose of document(see page 8)
- Purpose of REST API automated tests(see page 9)
- Prerequisites(see page 10)
- Structure(see page 11)
- Useful notes(see page 12)
- Test Case steps description(see page 13)
- Prerequisite test case(see page 14)
    - ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS(see page 14)
- Test Cases(see page 15)
    - CHECK_USING_NEW_SID(see page 15)
    - DEBIT_BET_ALREADY_EXIST(see page 15)
    - DEBIT_CANCEL(see page 15)
    - DEBIT_CREDIT(see page 16)
    - DEBIT_CREDIT_BET_ALREADY_SETTLED(see page 16)
    - DEBIT_CREDIT_BET_DOES_NOT_EXIST(see page 16)
    - DEBIT_CREDIT_WITH_DIFFERENT_SIDS(see page 17)
    - SAME_USER_PLAYING_WITH_DIFFERENT_CHANNEL_TYPE(see page 17)
    - TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS(see page 18)
    - VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME(see page 19)
    - CHECK_USING_INVALID_USER_ID(see page 19)
    - Verifies that non-existing user ID can't be used to perform requests.(see page 19)
    - DEBIT_CANCEL_BET_DOES_NOT_EXIST(see page 20)
    - DEBIT_CANCEL_BET_ALREADY_SETTLED(see page 21)
    - CREDIT_WITH_ZERO_AMOUNT(see page 21)
    - DEBIT_CREDIT_SETTLE_TYPE_GAMEWISE(see page 21)
    - MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL(see page 22)
    - MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT(see page 22)
    - INSUFFICIENT_FUNDS(see page 23)
    - ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL(see page 24)
    - CRAPS_DELAYED_SETTLEMENT_Long_Living_Bet(see page 24)
    - DEBIT_CREDIT_CANCEL(see page 25)
    - For casinos that don't use depositAfterCancel(see page 25)
    - DEBIT_CANCEL_CREDIT(see page 25)
    - DEBIT_CREDIT_WITH_BIG_PAYOUT(see page 25)
    - RESPONSE_TIME_VALIDATION(see page 26)
    - TIP_DEBIT_CANCEL(see page 26)
    - TIP_DEBIT_CLOSE(see page 27)
    - TIP_IDEMPOTENT_DEBIT(see page 27)
    - TIP_IDEMPOTENT_CANCEL(see page 27)
    - TIP_IDEMPOTENT_CLOSE(see page 28)
    - TIP_DEBIT_BETWEEN_GAMES(see page 28)
    - CANCEL_DEBIT_BEFORE_CANCEL_FOR_ANOTHER_GAME(see page 28)
    - FREE_ROUND_PLAYABLE_SPENT_PROMO_PAYOUT(see page 29)
    - DOUBLE_PROMO_PAYOUT(see page 29)
    - JACKPOT_PROMO_PAYOUT(see page 30)
    - FROM_GAME_PROMO_PAYOUT(see page 30)
    - REAL_TIME_MONETARY_REWARD_PROMO_PAYOUT(see page 30)
    - REWARD_GAME_MIN_BET_LIMIT_REACHED_PROMO_PAYOUT(see page 31)
    - REWARD_GAME_WIN_CAP_REACHED_PROMO_PAYOUT(see page 31)
    - REWARD_GAME_PLAYABLE_SPENT_PROMO_PAYOUT(see page 31)
    - NEW_TYPE_PROMO_PAYOUT(see page 32)
    - DEBIT_CREDIT_WITH_EVO/NE/RT_JACKPOT_STRUCTURE(see page 32)
    - DEBIT_CANCEL_WITH_EVO/NE/RT_JACKPOT_STRUCTURE(see page 32)

- PROMO_DEBIT_PROMO_CREDIT(see page 33)
- PROMO_DEBIT_PROMO_CANCEL(see page 33)
- DOUBLE_PROMO_DEBIT(see page 33)
- DOUBLE_PROMO_CREDIT(see page 34)
- DOUBLE_PROMO_CANCEL(see page 34)
- PROMO_CREDIT_NO_DEBIT(see page 34)
- PROMO_CANCEL_NO_DEBIT(see page 35)
- MULTIPLE_DEBIT_CREDIT_CANCEL_SETTLE_TYPE_MIXED(see page 35)
- DEBIT_CANCEL_BET_ALREADY_SETTLED_UNEXPECTED(see page 35)
- DEBIT_CREDIT_BET_ALREADY_SETTLED_UNEXPECTED(see page 36)

- AAMS Test Cases(see page 37)
  - AAMS_ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS(see page 37)
  - AAMS_SESSION_ALREADY_CLOSED(see page 37)
  - AAMS_SESSION_DOES_NOT_EXIST(see page 37)
  - AAMS_SESSION_ALREADY_CLOSED_CANCEL(see page 38)
  - AAMS_SESSION_ALREADY_CLOSED_CREDIT(see page 38)
  - AAMS_SESSION_ALREADY_CLOSED_DEBIT(see page 38)
  - AAMS_LOBBY_BALANCE(see page 39)
  - AAMS_DEBIT_CREDIT(see page 39)
  - AAMS_DEBIT_CANCEL(see page 39)
  - AAMS_DEBIT_CREDIT_CANCEL(see page 40)
  - AAMS_DEBIT_CANCEL_CREDIT(see page 40)
  - AAMS_DEBIT_BET_ALREADY_EXIST(see page 40)
  - AAMS_DEBIT_CREDIT_BET_DOES_NOT_EXIST(see page 41)
  - AAMS_DEBIT_CANCEL_BET_DOES_NOT_EXIST(see page 41)
  - AAMS_DEBIT_CREDIT_BET_ALREADY_SETTLED(see page 42)
  - AAMS_DEBIT_CANCEL_BET_ALREADY_SETTLED(see page 42)
  - AAMS_MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL(see page 43)
  - AAMS_MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT(see page 43)
  - AAMS_TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS(see page 44)
  - AAMS_ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL(see page 45)
  - AAMS_DEBIT_CREDIT_WITH_DIFFERENT_SIDS(see page 45)
  - AAMS_CHECK_USING_NEW_SID(see page 46)
  - AAMS_CREDIT_WITH_ZERO_AMOUNT(see page 46)
  - AAMS_DEBIT_CREDIT_SETTLE_TYPE_GAMEWISE(see page 47)
  - AAMS_VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME(see page 47)

1 Purpose of document

The purpose of the current document is to overview and explicitly guide through the Evolution Gaming REST API automated tests, which are an essential and one of the primary phases within Evolution and 3rd party integration process.

2 Purpose of REST API automated tests

The purpose of REST API automated tests is to verify the integration functionality by:

- calling REST API standard calls, i.e.: debit, credit, balance etc.;
- executing standard/happy scenarios, i.e.: bet placed, bet settled, balance updated;
- executing non-standard scenarios, i.e.: attempt to settle a bet, settlement of which had already succeeded;
- executing every case per user+game type (this means every case will be run separately per every provided user within every planned game type).

The integration is treated INCOMPLETE and NOT ready for LIVE before all REST API tests had passed

3 Prerequisites

- test environment network configuration (IPs whitelisted, endpoint provided);
- licensee side SID method implemented;
- licensee side debit, credit, cancel, balance calls support implemented (verified in ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS(see page 6));
- minimum 2 user accounts provided by licensee;
- user accounts balance >= 50;

4 Structure

Current description consists of

- test cases' steps definitions;
- test cases list, where every unit is to contain:
    - purpose of test case;
    - test case expected result;
    - test case steps.

5 Useful notes

- API test generate random table IDs.
- IF there is a predefined SID value, SID method is not called. General field name is "predefinedSid", but also there are additional fields for specific scenarios "secondaryPredefinedSid" for DEBIT_CREDIT_WITH_DIFFERENT_SIDS, "predefinedSidMobile" for SAME_USER_PLAYING_WITH_DIFFERENT_CHANNEL_TYPE. These sids are used instead of second sid requests.
- DEBIT_CREDIT_WITH_DIFFERENT_SIDS case is executed only IF it is allowed to change SID value within initialization process.
- Scenario runs for a first user from a config if scenario doesn't involve several users. In case the scenario needs to do calls with several users it would take first n-users from the config. Some scenarios(LOBBY_BALANCE) run for all the users from the config.
- By default 6 decimals will be send in amounts, but it is possible to override it in **decimalPoints** field.
- By default **all the scenarios** are executed only for first player and first game in the list, but there are **several exceptions**:
  - DEBIT_CREDIT executed for all combinations of (user, game)
  - DEBIT_CANCEL executed for all combinations of (user, game)
  - LOBBY_BALANCE executed for all users
- **List of the scenarios** that can be executed only with **a specific game**:
  - FIFTY_FIVE_DOND_SETTLEMENTS only with dealnodeal game
  - ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL only with craps or rng-craps game
  - DEBIT_CREDIT_WITH_DIFFERENT_SIDS only with blackjack game
  - MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT only with blackjack game
  - MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL only with blackjack game
  - TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS only with blackjack game and config should have at least 2 users

6 Test Case steps description

`Call service method SID()`

Evolution calls SID method to retrieve a SID value from licensee.

`Initialize.`

User authenticates in Evolution system.

`Place bet`

Evolution sends debit request to licensee system.

`End game`

Evolution sends credit request to licensee system.

`Check balance`

Evolution sends balance update request to licensee system.

`Session`

Session reference is AAMS (Italian regulator) specific session.

7 Prerequisite test case

7.1 ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS

Verification of main wallet methods before running specific test cases/scenarios. Main wallet methods are: get balance, place bet, settle bet.

Expected behaviour: get balance, place bet and credit the bet, check balance changes.

```
Call service method sid(). Get new SID.
Initialize (method check()). Verify response - response must contain status OK.
Check balance (method balance()). Verify response - response must contain status
OK.
Place bet (method debit() with default amount). Verify response - response must
contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8 Test Cases

8.1 CHECK_USING_NEW_SID

Verification of incorrect SID supply case: if a new SID returned after initialization, further requests storing old SID must be rejected.

Expected behavior: Requests with old SID should be rejected.

```
Call service method SID(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
If after initialize response contains new SID, must verify whether old SID can
be used.
Check balance with old SID. Verify response - response must contain failures
(status INVALID_SID).
Place bet with old SID (method debit() with default amount). Verify response -
response must contain failures (status INVALID_SID).
End game with old SID (method credit() with default amount). Verify response -
response must contain failures (status INVALID_SID or BET_DOES_NOT_EXIST).
```

8.2 DEBIT_BET_ALREADY_EXIST

Verifies that bet can be handled out only once.

Expected behavior: Second request to place same bet should be rejected.

```
Call service method SID(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Place bet with the same refId (see previous step). Verify response - response
must contain failures (status BET_ALREADY_EXIST or OK). Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.3 DEBIT_CANCEL

Verifies that bet can be canceled and balance changes appropriately.

Expected behavior: Cancel request should be proceeded correctly.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
```

```
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Cancel bet (with the same transactionId(see previous step)). Verify response -
response must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.4 DEBIT_CREDIT

Verifies that bet can be placed, paid out and balance changes appropriately.

Expected behavior: Credit request should be proceeded correctly.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.5 DEBIT_CREDIT_BET_ALREADY_SETTLED

Verifies that bet can be payed out only once. Attempts to payout for same bet should be rejected and contain error BET_ALREADY_SETTLED.

Expected behavior: Second request to payout for same bet should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
End game (method credit() with default amount) with the same refId(see previous
step). Verify response - response must contain failures (status
BET_ALREADY_SETTLED or OK). Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.6 DEBIT_CREDIT_BET_DOES_NOT_EXIST

Verifies if credit does not succeed for a bet with non-existing reference ID and BET_DOES_NOT_EXIST response is the case. Random - and must not be accepted/processed

UNEXPECTED scenario in terms of Evolution.

Expected behavior: Request with fake reference ID should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game with fake refId. Verify response - response must contain failures
(status BET_DOES_NOT_EXIST). Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
End game (method credit() call). Verify response - response must contain status
OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.7 DEBIT_CREDIT_WITH_DIFFERENT_SIDS

Verifies if separate bets placed within different SIDs are processed successfully.

Expected behavior: All requests should be proceeded successfully.

```
Runs for games with Multi-transaction support and Mixed settlement type.
Call service method sid(). Get first SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet for first SID. Verify response - response must contain status
OK. Verify balance.
Call service method sid(). Get second SID. Verify response - first and second
SID must be different.
Initialize. Verify response - response must contain status OK.
Place bet for another second SID. Verify response - response must contain status
OK. Verify balance.
End game (method credit() call) for both bets with different SIDs. Verify
response - response must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.8 SAME_USER_PLAYING_WITH_DIFFERENT_CHANNEL_TYPE

Verifies if separate bets placed within different sids and channels are processed successfully.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid() with channelType = 'P'. Getting new SID('sid_A').
Initialize (method check() call) with channelType = 'P'. Verify response -
response must contain status OK.
Call service method sid() with channelType = 'M'. Getting new SID('sid_B').
```

```
Initialize with channelType = 'M'. Verify response - response must contain
status OK.
Check balance for sid = 'sid_A'. Verify response - response must contain status
OK.
Check balance for sid = 'sid_B'. Verify response - response must contain status
OK.
Place bet for sid = 'sid_A'. Verify response - response must contain status OK.
Verify balance.
Place bet for sid = 'sid_B'. Verify response - response must contain status OK.
Verify balance.
End game (method credit() call) for both bets with different sids. Verify
response - response must contain status OK. Verify balance.
Check balance for sid = 'sid_A'. Verify response - response must contain status
OK. Verify balance.
Check balance for sid = 'sid_B'. Verify response - response must contain status
OK. Verify balance.
```

8.9 TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS

Verifies if multiple (2) bets placed by different users at a time are processed successfully.

Expected behavior: All requests should be proceeded successfully.

```
Runs for games with Multi-transaction support when at least two users are
defined.
Call service method sid() for userId = 'a'. Getting new SID('sid_A').
Call service method sid() for userId = 'b'. Getting new SID('sid_B').
Initialize (method check() call) for userId = 'a'. Verify response - response
must contain status OK.
Initialize (method check() call) for userId = 'b'. Verify response - response
must contain status OK.
Check balance for userId = 'a'. Verify response - response must contain status
OK.
Check balance for userId = 'b'. Verify response - response must contain status
OK.
Place bet for userId = 'a'. Verify response - response must contain status OK.
Verify balance.
Place bet for userId = 'b'. Verify response - response must contain status OK.
Verify balance.
Place bet for userId = 'a'. Verify response - response must contain status OK.
Verify balance.
Place bet for userId = 'b'. Verify response - response must contain status OK.
Verify balance.
End game (method credit() call) for both bets (userId = 'a'). Verify response -
response must contain status OK. Verify balance.
End game (method credit() call) for both bets (userId = 'b'). Verify response -
response must contain status OK. Verify balance.
Check balance for userId = 'a'. Verify response - response must contain status
OK. Verify balance.
Check balance for userId = 'b'. Verify response - response must contain status
OK. Verify balance.
```

• 8.10 VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME

Verifies if user A can place a bet before user's A previous bet had been settled.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet for game id = 'a'. Verify response - response must contain status OK.
Verify balance.
Place bet for another game id = 'b'. Verify response - response must contain
status OK. Verify balance.
End game (method credit() call) for game id = 'b'. Verify response - response
must contain status OK. Verify balance.
End game (method credit() call) for game id = 'a'. Verify response - response
must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

- **LOBBY_BALANCE**

Verifies balance update within Evolution Lobby/out of a game. This scenario runs for all the users from the config.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check lobby balance (game is null). Verify response - response must contain
status OK.
```

• 8.11 CHECK_USING_INVALID_USER_ID

8.12 Verifies that non-existing user ID can't be used to perform requests.

Expected behavior: All requests with non-existing user ID should be rejected.

```
Call service method sid(). Getting new SID.
Put in requests userId = 'fakeUserId'. Initialize (method check() call). Verify
```

```
response - response must contain status INVALID_PARAMETER.
Put in requests correct userId value.  Initialize (method check() call). Verify
response - response must contain status OK.
Put in requests userId = 'fakeUserId'. Check balance. Verify response - response
must contain status INVALID_PARAMETER.
Put in requests correct userId value.  Check balance. Verify response - response
must contain status OK.
Put in requests userId = 'fakeUserId'. Place bet. Verify response - response
must contain status INVALID_PARAMETER. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
Put in requests correct userId value.  Place bet. Verify response - response
must contain status OK. Verify balance.
Put in requests userId = 'fakeUserId'. Cancel Bet. Verify response - response
must contain status INVALID_PARAMETER. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
Put in requests userId = 'fakeUserId'. Credit Bet. Verify response - response
must contain status INVALID_PARAMETER. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
Put in requests correct userId value.  Credit Bet. Verify response - response
must contain status OK. Verify balance.
```

Check balance. Verify response - response must contain status OK. `Verify balance.`

8.13 DEBIT_CANCEL_BET_DOES_NOT_EXIST

Verifies if cancel does not succeed for a bet with non-existing reference ID and BET_DOES_NOT_EXIST response is the case

UNEXPECTED scenario in terms of Evolution

Expected behavior: Request with non-existing reference ID should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
```

Place bet. Verify response - response must contain status OK. `Verify balance.`

Cancel bet with fake transactionId and refId. Verify response - response must contain failures (status BET_DOES_NOT_EXIST). `Verify balance.`

Check balance. Verify response - response must contain status OK. `Verify balance.`

Cancel bet. Verify response - response must contain status OK. `Verify balance.`

Check balance. Verify response - response must contain status OK. `Verify balance.`

8.14 DEBIT_CANCEL_BET_ALREADY_SETTLED

Verifies that bet can be canceled only once.

Expected behavior: Second request for bet cancel should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Cancel bet. Verify response - response must contain status OK. Verify balance.
Cancel bet with the same transactionId and refId(see previous step). Verify
response - response must contains failures (status BET_ALREADY_SETTLED or OK).
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.15 CREDIT_WITH_ZERO_AMOUNT

Verification of 0 amount payout processing.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() call with amount 0). Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.16 DEBIT_CREDIT_SETTLE_TYPE_~~GAMEWISE~~

Verifies that double payout for the same game wont be performed.

Expected behavior: Second request with payout should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet for game id = 'a'. Verify response - response must contain status OK.
Verify balance.
Place bet for game id = 'a'. Verify response - response must contain status OK.
Verify balance.
End game (method credit() call) for game id = 'a'. Verify response - response
must contain status OK. Verify balance.
End game (method credit() call) for game id = 'a'. Verify response - response
must contains failures (status BET_ALREADY_SETTLED or OK). Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

By default must be applied Mixed

8.17 MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL

For casinos that don't use depositAfterCancel.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place initial bet. Verify response - response must contain status OK. Verify
balance.
Place next bet. Verify response - response must contain status OK. Verify
balance.
Settle initial bet. Verify response - response must contain status OK. Verify
balance.
Cancel next bet. Verify response - response must contain status OK. Verify
balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.18 MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT

Verifies that different bets can be canceled and settled sequentially.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place initial bet. Verify response - response must contain status OK. Verify
balance.
Place next bet. Verify response - response must contain status OK. Verify
balance.
Cancel next bet. Verify response - response must contain status OK. Verify
balance.
Settle initial bet. Verify response - response must contain status OK. Verify
balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

• 8.19 INSUFFICIENT_FUNDS

Bet amount is bigger than balance. If balance returned in response with error code INSUFFICIENT_FUNDS, it must be same as was for 1st balance call.

Note: If stub-server is being used, set performance.test.mode=off, otherwise you will get infinite balance.

Expected behavior: Balance value should be the same in both check balance requests.

```
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place initial bet. Verify response - response must contain status
INSUFFICIENT_FUNDS. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

- ~~**FIFTY_FIVE_DOND_SETTLEMENTS_SCENARIO**~~ **MULTIPLE_DOND_SETTLEMENTS_SCENARIO**

Verification of user making from 1 to 101(by default 101) top-up bets in round. (Transactions quantity configuration occurs in the config).

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() call). Verify response - response must contain status
OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.20 ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL

Executed for casinos that don't use depositAfterCancel

Expected behavior: Second cancel bet request should be canceled.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet for game id = 'a'. Verify response - response must contain status
OK. Verify balance.
Place bet for game id = 'b'. Verify response - response must contain status
OK. Verify balance.
Place bet for game id = 'b'. Verify response - response must contain status
OK. Verify balance.
Place bet for game id = 'a'. Verify response - response must contain status
OK. Verify balance.
Cancel 2nd bet for game id = 'b'. Verify response - response must contain status
OK. Verify balance.
End game (method credit() call) for game id = 'a'. Verify response - response
must contain status OK. Verify balance.
End game (method credit() call) for game id = 'b'. Verify response - response
must contain status OK. Verify balance.
Cancel 1st bet for game id = 'b'. Verify response - response must contain
failures (status BET_ALREADY_SETTLED). Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.21 CRAPS_DELAYED_SETTLEMENT_Long_Living_Bet

`Test runs in 2 parts: first we put a bet, then new job is generated that settles the bet after 12h.`

Expected behavior: All requests should be proceeded successfully.

```
1st part:
Call service method sid().
Getting new SID. Initialize. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK.
Place delayed bet. Verify response - response must contain status OK. Verify
balance.
Check balance. Verify response - response must contain status OK. Verify
balance.

2nd part:
Call service method sid().
Getting new SID. Initialize. Verify response - response must contain status OK.
```

```
Check balance. Verify response - response must contain status OK.
Settle delayed bet. Verify response - response must contain status OK. Verify
balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.22 DEBIT_CREDIT_CANCEL

8.23 For casinos that don't use depositAfterCancel

Expected behavior: Cancel bet request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place Bet. Verify response - response must contain status OK. Verify Balance.
Settle Bet. Verify response - response must contain status OK. Verify Balance.
Cancel Bet. Verify response - response must contain status BET_ALREADY_SETTLED.
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.24 DEBIT_CANCEL_CREDIT

Verifies that bet cant be settled after cancelation.

Expected behavior: Bet payout request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place Bet. Verify response - response must contain status OK. Verify Balance.
Cancel Bet. Verify response - response must contain status OK. Verify Balance.
Settle Bet. Verify response - response must contain status BET_ALREADY_SETTLED.
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.25 DEBIT_CREDIT_WITH_BIG_PAYOUT

Verifies that big payouts can be processed.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place Bet. Verify response - response must contain status OK. Verify Balance.
End game with big amount 63727092.340000. Verify response - response must
contain status OK. Verify Balance.
Place Bet with big amount 63727092.340000. Verify response - response must
contain status OK. Verify Balance.
End game (method credit() call). Verify response - response must contain status
OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.26 RESPONSE_TIME_VALIDATION

Verification of user making 10 top-up bets in round. Verification of response time.

Expected behavior: All requests should be proceeded successfully and response time must be less or equal than 750 millis.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
```

`Check balance. Verify response - response must contain status OK. Verify response time.`

`Place bet. Verify response - response must contain status OK. Verify balance. Verify response time.`

`End game (method credit() call). Verify response - response must contain status OK. Verify balance. Verify response time.`

`Check balance. Verify response - response must contain status OK. Verify balance. Verify response time.`

8.27 TIP_DEBIT_CANCEL

Verification of tip cancelation.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Debit Tip. Verify response - response must contain status OK. Verify Balance.
```

```
Cancel Tip. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.28 TIP_DEBIT_CLOSE

Verifies that tip can be closed.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Debit Tip. Verify response - response must contain status OK. Verify Balance.
Close Tip. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.29 TIP_IDEMPOTENT_DEBIT

Verifies that idempotent tip can be closed.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Debit Tip. Verify response - response must contain status OK. Verify Balance.
Debit Tip(The same operation with different uuid). Verify response - response
must contain status OK. Verify Balance.
Close Tip. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.30 TIP_IDEMPOTENT_CANCEL

Verifies that tip can be idempotently canceled.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Debit Tip. Verify response - response must contain status OK. Verify Balance.
```

```
Cancel Tip. Verify response - response must contain status OK. Verify Balance.
Cancel Tip (the same operation with different uuid). Verify response - response
must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.31 TIP_IDEMPOTENT_CLOSE

Verifies that tip can be idempotently closed.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Debit Tip. Verify response - response must contain status OK. Verify Balance.
Close Tip. Verify response - response must contain status OK. Verify Balance.
Close Tip (the same operation with different uuid). Verify response - response
must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.32 TIP_DEBIT_BETWEEN_GAMES

Verifies that tips between games can be closed.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Debit Tip. Verify response - response must contain status OK. Verify Balance.
Close Tip. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.33 CANCEL_DEBIT_BEFORE_CANCEL_FOR_ANOTHER_GAME

To verify can integration have settings to not retry bet and send cancel instead. Cancel to be remembered and bet with same transaction id must not be accepted.

Expected behavior: Requests with incorrect game ID should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
```

```
OK.
Check balance. Verify response - response must contain status OK.
Place Bet for gameId = 'a'. Verify response - response must contain status OK.
Verify Balance.
Cancel Bet for gameId = 'b'. Verify response - response must contain status
BET_DOES_NOT_EXIST. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
Place Bet for gameId = 'b'. Verify response - response must contain status
FINAL_ERROR_ACTION_FAILED. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
Cancel Bet for gameId = 'a'. Verify response - response must contain status OK.
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

specific for integration - don't retry but send cancel

can be unhappy - for all - if no need in such behaviour

8.34 FREE_ROUND_PLAYABLE_SPENT_PROMO_PAYOUT

Verification of FreeRoundPlayableSpent promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. FreeRoundPlayableSpent type. Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.35 DOUBLE_PROMO_PAYOUT

Verifies that double promo payouts won't be processed.

Expected behavior: Second payout request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. Verify response - response must contain status OK. Verify balance.
Promo payout. With the same transaction id. Verify response - response must
contain status OK or BET_ALREADY_SETTLED. Verify balance.
```

```
Check balance. Verify response - response must contain status OK. Balance should
be changed only after the first transaction.
```

8.36 JACKPOT_PROMO_PAYOUT

Verification of Jackpot promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK.
Promo payout. JackpotWin type. Verify response - response must contain status
OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.37 FROM_GAME_PROMO_PAYOUT

Verification of FromGame promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. FromGame type. Verify response - response must contain status OK.
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.38 REAL_TIME_MONETARY_REWARD_PROMO_PAYOUT

Verification of RealTimeMonetaryReward promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. RealTimeMonetaryReward type. Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.39 REWARD_GAME_MIN_BET_LIMIT_REACHED_PROMO_PAYOUT

Verification of RewardGameMinBetLimitReached promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. RewardGameMinBetLimitReached type. Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.40 REWARD_GAME_WIN_CAP_REACHED_PROMO_PAYOUT

Verification of RewardGameWinCapReached promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. RewardGameWinCapReached type. Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.41 REWARD_GAME_PLAYABLE_SPENT_PROMO_PAYOUT

Verification of RewardGamePlayableSpent promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. RewardGamePlayableSpent type. Verify response - response must
contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.42 NEW_TYPE_PROMO_PAYOUT

Verification of NewType promo payout handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo payout. NewType type. Verify response - response must contain status OK.
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.43 DEBIT_CREDIT_WITH_EVO/NE/RT_JACKPOT_STRUCTURE

Verification of EVO/NE/RT jackpot format bet settlement handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet(debit with EVO/NE/RT jackpot format). Verify response - response must
contain status OK. Verify balance.
Settle Bet(credit with EVO/NE/RT jackpot format). Verify response - response
must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.44 DEBIT_CANCEL_WITH_EVO/NE/RT_JACKPOT_STRUCTURE

Verification of EVO/NE/RT jackpot format bet cancelation handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
Balance.
Place bet (debit with EVO/NE/RT jackpot format). Verify response - response must
contain status OK. Verify balance.
Cancel Bet (cancel with EVO/NE/RT jackpot format). Verify response - response
must contain status OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.45 PROMO_DEBIT_PROMO_CREDIT

Verification of promo bet settlement handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo debit. Verify response - response must contain status OK. Verify Balance.
Promo credit. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.46 PROMO_DEBIT_PROMO_CANCEL

Verification of promo bet cancelation handling.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo debit. Verify response - response must contain status OK. Verify Balance.
Promo cancel. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.47 DOUBLE_PROMO_DEBIT

Verifies that double promo bet wont be processed.

Expected behavior: Second bet request should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo debit. Verify response - response must contain status OK. Verify Balance.
Promo debit. Verify response - response must contain status BET_ALREADY_EXISTS
or OK. Verify balance.
Promo credit. Verify response - response must contain status OK. Verify Balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.48 DOUBLE_PROMO_CREDIT

Verifies that promo bet wont be settled twice.

Expected behavior: Second bet payout request should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo debit. Verify response - response must contain status OK. Verify Balance.
Promo credit. Verify response - response must contain status OK. Verify Balance.
Promo credit. Verify response - response must contain status BET_ALREADY_SETTLED
or OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.49 DOUBLE_PROMO_CANCEL

Verifies that promo bet wont be canceled twice.

Expected behavior: Second bet cancel request should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo debit. Verify response - response must contain status OK. Verify Balance.
Promo cancel. Verify response - response must contain status OK. Verify Balance.
Promo cancel with same refId. Verify response - response must contain status
BET_ALREADY_SETTLED or OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.50 PROMO_CREDIT_NO_DEBIT

Verifies that not existing promo bet wont be settled.

Expected behavior: Request with payout should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo credit. Verify response - response must contain status BET_DOES_NOT_EXIST
or OK. Verify balance.
```

```
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.51 PROMO_CANCEL_NO_DEBIT

Verifies that not existing promo bet wont be canceled.

Expected behavior: Cancel bet request should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Promo cancel. Verify response - response must contain status BET_DOES_NOT_EXIST
or OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

8.52 MULTIPLE_DEBIT_CREDIT_CANCEL_SETTLE_TYPE_MIXED

Verifies the bet handling process in mixed type with multiple bets.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place Bet 15 times. Verify response - response must contain status OK. Verify
balance.
End game (method credit() call) for 1-7 bets with x2 debit amount. Verify
response - response must contain status OK. Verify balance.
End game (method credit() call) for 8-11 bets with debit amount. Verify response
- response must contain status OK. Verify balance.
Cancel game(12-15 bets). Verify response - response must contain status OK.
Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.53 DEBIT_CANCEL_BET_ALREADY_SETTLED_UNEXPECTED

Verifies that bet with incorrect transactionId wont be canceled.

UNEXPECTED scenario in terms of Evolution

Expected behavior: Second cancel bet request should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Cancel bet. Verify response - response must contain status OK. Verify balance.
Cancel bet with the different transactionId (see previous step). Verify response
- response must contains failures (status BET_ALREADY_SETTLED) or OK. Verify
balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

8.54 DEBIT_CREDIT_BET_ALREADY_SETTLED_UNEXPECTED

Verifies that bet with same transactionId wont be settled second time.

UNEXPECTED scenario in terms of Evolution

Expected behavior: Second payout request should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
End game (method credit() with default amount) with the different
transactionId(see previous step). Verify response - response must contain
failures (status BET_ALREADY_SETTLED) or OK. Verify balance.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9 AAMS Test Cases

9.1 AAMS_ALL_METHODS_CHECK_BEFORE_RUNNING_TESTS

Verification of main wallet methods before running all the test cases/scenarios.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check()). Verify response - response must contain status OK.
Check balance (method balance()). Verify response - response must contain status
OK.
Call open session. Verify response - response must contain status OK.
Place bet (method debit() with default amount). Verify response - response must
contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.2 AAMS_SESSION_ALREADY_CLOSED

Purpose of the test is verification of the AAMS session opening and closure.

Expected behavior: Second session close request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response -  response must contain status OK.
Call close session. Verify response - response must contain status OK.
Call close session. Verify response - response must contain status
SESSION_ALREADY_CLOSED or OK.
Check balance. Verify response - response must contain status OK.
```

9.3 AAMS_SESSION_DOES_NOT_EXIST

Purpose of the test is verification of not existing AAMS session handling.

Expected behavior: Session close request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Call close session. Verify response - response must contain status
SESSION_DOES_NOT_EXIST.
```

```
Check balance. Verify response - response must contain status OK.
```

9.4 AAMS_SESSION_ALREADY_CLOSED_CANCEL

Purpose of the test is verification that after the AAMS session is closed, the bets made within the session are not available.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response -  response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Cancel bet. Verify response - response must contain failures
SESSION_ALREADY_CLOSED. Verify balance.
Check balance. Verify response - response must contain status OK.
```

9.5 AAMS_SESSION_ALREADY_CLOSED_CREDIT

Purpose of the test is verification that after the AAMS session is closed, the bets made within the session are not available.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response -  response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
End game(method credit() with default amount). Verify response - response must
contain failures SESSION_ALREADY_CLOSED. Verify balance.
Check balance. Verify response - response must contain status OK.
```

9.6 AAMS_SESSION_ALREADY_CLOSED_DEBIT

Purpose of the test is verification that after the AAMS session is closed, the bets made within the session are not available.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response -  response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
```

```
Place bet. Verify response - response must contain failures
SESSION_ALREADY_CLOSED. Verify balance.
Check balance. Verify response - response must contain status OK.
```

9.7 AAMS_LOBBY_BALANCE

Verifies balance update within Evolution Lobby/out of a game. This scenario runs for all the users from the config.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check lobby balance (game is null). Verify response - response must contain
status OK.
```

9.8 AAMS_DEBIT_CREDIT

Verification of a bet life-cycle: bet is placed and paid out.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.9 AAMS_DEBIT_CANCEL

Verification of bet cancellation and respective balance update.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Cancel bet (with the same transactionId(see previous step)). Verify response -
```

```
response must contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.10 AAMS_DEBIT_CREDIT_CANCEL

For casinos that don't use depositAfterCancel

Expected behavior: Cancel bet request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place Bet. Verify response - response must contain status OK. Verify Balance.
Settle Bet. Verify response - response must contain status OK. Verify Balance.
Cancel Bet. Verify response - response must contain status BET_ALREADY_SETTLED.
Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

9.11 AAMS_DEBIT_CANCEL_CREDIT

Verifies that bet cant be settled after cancelation.

Expected behavior: Payout bet request should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place Bet. Verify response - response must contain status OK. Verify Balance.
Cancel Bet. Verify response - response must contain status OK. Verify Balance.
Settle Bet. Verify response - response must contain status BET_ALREADY_SETTLED.
Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
Balance.
```

9.12 AAMS_DEBIT_BET_ALREADY_EXIST

Verifies if a unique transaction ID can be debited only once, otherwise BET_ALREADY_EXIST response must be the case.

Expected behavior: Second request to place same bet should be rejected.

```
Call service method SID(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
Place bet with the same refId (see previous step). Verify response - response
must contain failures (status BET_ALREADY_EXIST or OK). Verify balance.
End game (method credit() with default amount). Verify response - response must
contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.13 AAMS_DEBIT_CREDIT_BET_DOES_NOT_EXIST

Verifies if credit does not succeed for a bet with non-existing reference ID and BET_DOES_NOT_EXIST response is the case. Random - and must not be accepted/processed

UNEXPECTED scenario in terms of Evolution

Expected behavior: Request with fake reference ID should be rejected.

```
Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game with fake refId. Verify response - response must contain failures
(status BET_DOES_NOT_EXIST). Verify balance.
End game (method credit() call). Verify response - response must contain status
OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.14 AAMS_DEBIT_CANCEL_BET_DOES_NOT_EXIST

Verifies if cancel does not succeed for a bet with non-existing reference ID and BET_DOES_NOT_EXIST response is the case

UNEXPECTED scenario in terms of Evolution

Expected behavior: Request with non-existing reference ID should be rejected.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
```

Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. `Verify balance.`
Cancel bet with fake transactionId and refId. Verify response - response must contain failures (status BET_DOES_NOT_EXIST). `Verify balance.`
Cancel bet. Verify response - response must contain status OK. `Verify balance.`
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. `Verify balance.`

9.15 AAMS_DEBIT_CREDIT_BET_ALREADY_SETTLED

Verifies if a unique transaction ID can be credited only once, otherwise BET_ALREADY_SETTLED response must be the case.

Expected behavior: Second request to payout for same bet should be rejected.

Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. Verify balance.
End game (method credit() with default amount). Verify response - response must contain status OK. Verify balance.
End game (method credit() with default amount) with the same refId(see previous step). Verify response - response must contain failures (status BET_ALREADY_SETTLED or OK). Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify balance.

9.16 AAMS_DEBIT_CANCEL_BET_ALREADY_SETTLED

Verifies that second attempt of bet cancelation handled properly.

Expected behavior: Second request for bet cancel should be rejected.

Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet. Verify response - response must contain status OK. `Verify balance.`
Cancel bet. Verify response - response must contain status OK. `Verify balance.`
Cancel bet with the same transactionId and refId(see previous step). Verify response - response must contains failures (status BET_ALREADY_SETTLED or OK).

`Verify balance.`
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. `Verify balance.`

9.17 AAMS_MULTI_STEP_GAME_TWO_DEBITS_CREDIT_CANCEL

For casinos that don't use depositAfterCancel.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place initial bet. Verify response - response must contain status OK. Verify
balance.
Place next bet. Verify response - response must contain status OK. Verify
balance.
Settle initial bet. Verify response - response must contain status OK. Verify
balance.
Cancel next bet. Verify response - response must contain status OK. Verify
balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.18 AAMS_MULTI_STEP_GAME_TWO_DEBITS_CANCEL_CREDIT

Verifies that different bets can be canceled and settled sequentially.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place initial bet. Verify response - response must contain status OK. Verify
balance.
Place next bet. Verify response - response must contain status OK. Verify
balance.
Cancel next bet. Verify response - response must contain status OK. Verify
balance.
Settle initial bet. Verify response - response must contain status OK. Verify
balance.
```

```
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.19 AAMS_TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS

Verifies if multiple (2) bets placed by different users at a time are processed successfully.

Expected behavior: All requests should be proceeded successfully.

```
Runs for games with Multi-transaction support when at least two users are
defined.
Call service method sid() for userId = 'a'. Getting new SID('sid_A').
Call service method sid() for userId = 'b'. Getting new SID('sid_B').
Initialize (method check() call) for userId = 'a'. Verify response - response
must contain status OK.
Initialize (method check() call) for userId = 'b'. Verify response - response
must contain status OK.
Check balance for userId = 'a'. Verify response - response must contain status
OK.
Check balance for userId = 'b'. Verify response - response must contain status
OK.
Call open session for userId = 'a'. Verify response - response must contain
status OK.
Call open session for userId = 'b'. Verify response - response must contain
status OK.
Place bet for userId = 'a'. Verify response - response must contain status OK.
Verify balance.
Place bet for userId = 'b'. Verify response - response must contain status OK.
Verify balance.
Place bet for userId = 'a'. Verify response - response must contain status OK.
Verify balance.
Place bet for userId = 'b'. Verify response - response must contain status OK.
Verify balance.
End game (method credit() call) for both bets (userId = 'a'). Verify response -
response must contain status OK. Verify balance.
End game (method credit() call) for both bets (userId = 'b'). Verify response -
response must contain status OK. Verify balance.
Call open session for userId = 'a'. Verify response - response must contain
status OK.
Call open session for userId = 'b'. Verify response - response must contain
status OK.
Check balance for userId = 'a'. Verify response - response must contain status
OK. Verify balance.
Check balance for userId = 'b'. Verify response - response must contain status
OK. Verify balance.
```

9.20 AAMS_ONE_PLAYER_PLAYS_TWO_GAME_ROUNDS_ON_ONE_TABLE_IN_PARALLEL

Executed for casinos that don't use depositAfterCancel

Expected behavior: Second cancel bet request should be canceled.

```
Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet for game id = 'a'. Verify response - response must contain status
OK. Verify balance.
Place bet for game id = 'b'. Verify response - response must contain status
OK. Verify balance.
Place bet for game id = 'b'. Verify response - response must contain status
OK. Verify balance.
Place bet for game id = 'a'. Verify response - response must contain status
OK. Verify balance.
Cancel 2nd bet for game id = 'b'. Verify response - response must contain status
OK. Verify balance.
End game (method credit() call) for game id = 'a'. Verify response - response
must contain status OK. Verify balance.
End game (method credit() call) for game id = 'b'. Verify response - response
must contain status OK. Verify balance.
Cancel 1st bet for game id = 'b'. Verify response - response must contain
failures (status BET_ALREADY_SETTLED). Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.21 AAMS_DEBIT_CREDIT_WITH_DIFFERENT_SIDS

Verifies if separate bets placed within different SIDs are processed successfully.

Expected behavior: All requests should be proceeded successfully.

```
Runs for games with Multi-transaction support and Mixed settlement type.
Call service method sid(). Get first SID.
Initialize (method check() call). Verify response - response must contain status
OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet for first SID. Verify response - response must contain status
OK. Verify balance.
Call service method sid(). Get second SID. Verify response - first and second
SID must be different.
Initialize. Verify response - response must contain status OK.
Place bet for another second SID. Verify response - response must contain status
```

```
OK. Verify balance.
End game (method credit() call) for both bets with different SIDs. Verify
response - response must contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.22 AAMS_CHECK_USING_NEW_SID

Verification of incorrect SID supply case: if a new SID returned after initialization, further requests storing old SID must be rejected.

Expected behavior: Requests with old SID should be rejected.

```
Call service method SID(). Get new SID.
Initialize (method check() call). Verify response - response must contain status
OK.
If after initialize response contains new SID, must verify whether old SID can
be used.
Check balance with old SID. Verify response - response must contain failures
(status INVALID_SID).
Call open session. Verify response - response must contain status OK.
Place bet (method debit() with default amount). Verify response - response must
contain failures (status INVALID_SID).
End game (method credit() with default amount). Verify response - response must
contain failures (status INVALID_SID or BET_DOES_NOT_EXIST).
Call close session. Verify response - response must contain status OK.
```

9.23 AAMS_CREDIT_WITH_ZERO_AMOUNT

Verification of 0 amount payout processing.

Expected behavior: All requests should be proceeded successfully.

```
Call service method sid(). Getting new SID.
Initialize (method check() call).  Verify response - response must contain
status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK. Place bet.
Verify response - response must contain status OK. Verify balance.
End game (method credit() call with amount 0). Verify response - response must
contain status OK. Verify balance.
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. Verify
balance.
```

9.24 AAMS_DEBIT_CREDIT_SETTLE_TYPE_~~GAMEWISE~~

Purpose of the test is verify that double payout for the same game wont be performed.

Expected behavior: Second request with payout should be rejected.

Call service method sid(). Getting new SID.
Initialize (method check() call). Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet for game id = 'a'. Verify response - response must contain status OK. `Verify balance.`
Place bet for game id = 'a'. Verify response - response must contain status OK. `Verify balance.`
End game (method credit() call) for game id = 'a'. Verify response - response must contain status OK. `Verify balance.`
End game (method credit() call) for game id = 'a'. Verify response - response must contains failures (status BET_ALREADY_SETTLED or OK). `Verify balance.`
Call close session. Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK. `Verify balance.`

9.25 AAMS_VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME

Verifies if user A can place a bet before user's A previous bet had been settled.

Purpose of the test is verification of that the user can make a bet in one game and another bet in a different game before the payment of the first bet.

Expected behaviour: It is possible to make a bet for one game and for the second game.

Call service method sid(). Get new SID.
Initialize (method check() call). Verify response - response must contain status OK.
Check balance. Verify response - response must contain status OK.
Call open session. Verify response - response must contain status OK.
Place bet for game id = 'a'. Verify response - response must contain status OK. `Verify balance.`
Place bet for another game id = 'b'. Verify response - response must contain status OK. `Verify balance.`
End game (method credit() call) for game id = 'b'. Verify response - response must contain status OK. `Verify balance.`
End game (method credit() call) for game id = 'a'. Verify response - response must contain status OK. `Verify balance.`

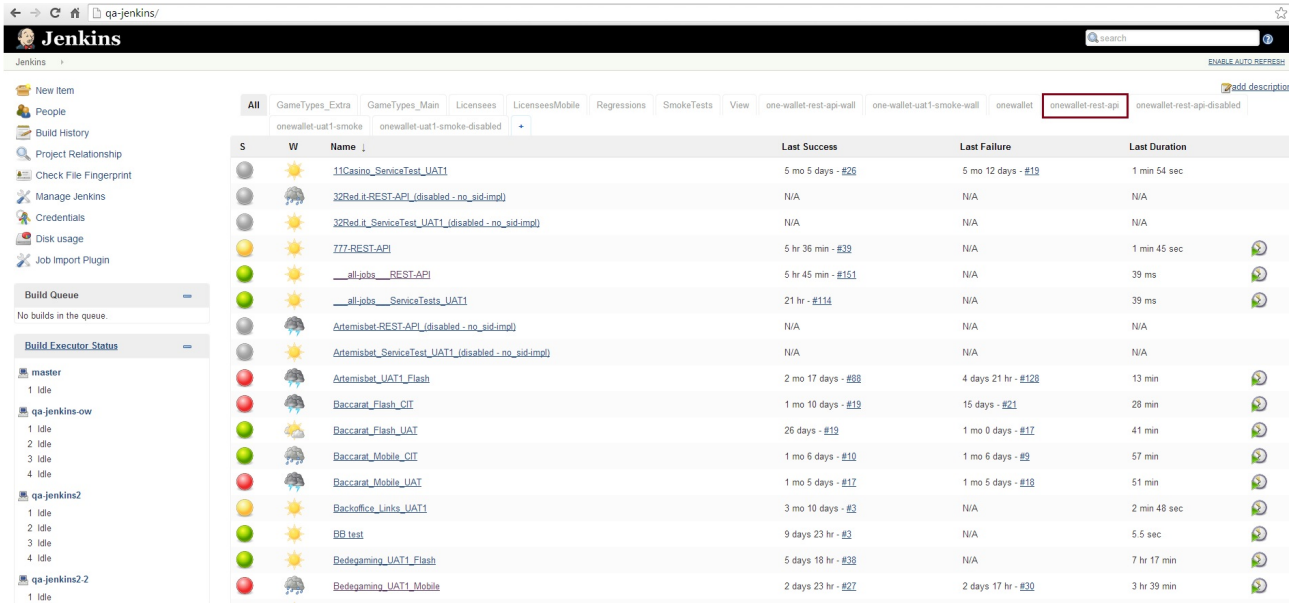
`Call close session. Verify response - response must contain status OK. Check balance. Verify response - response must contain status OK. Verify balance.`

10 REST API tests results

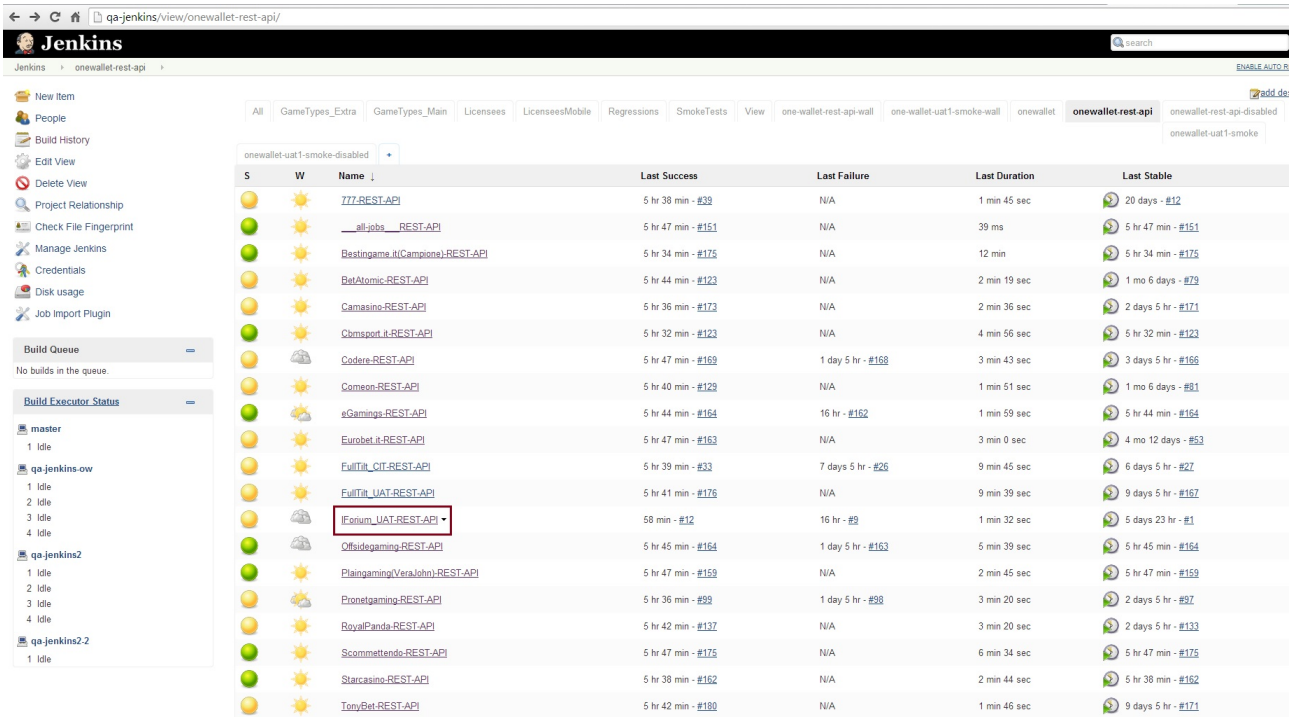
> Follow the below steps in order to review a particular test result/ output.

1. Go to http://qa-jenkins/

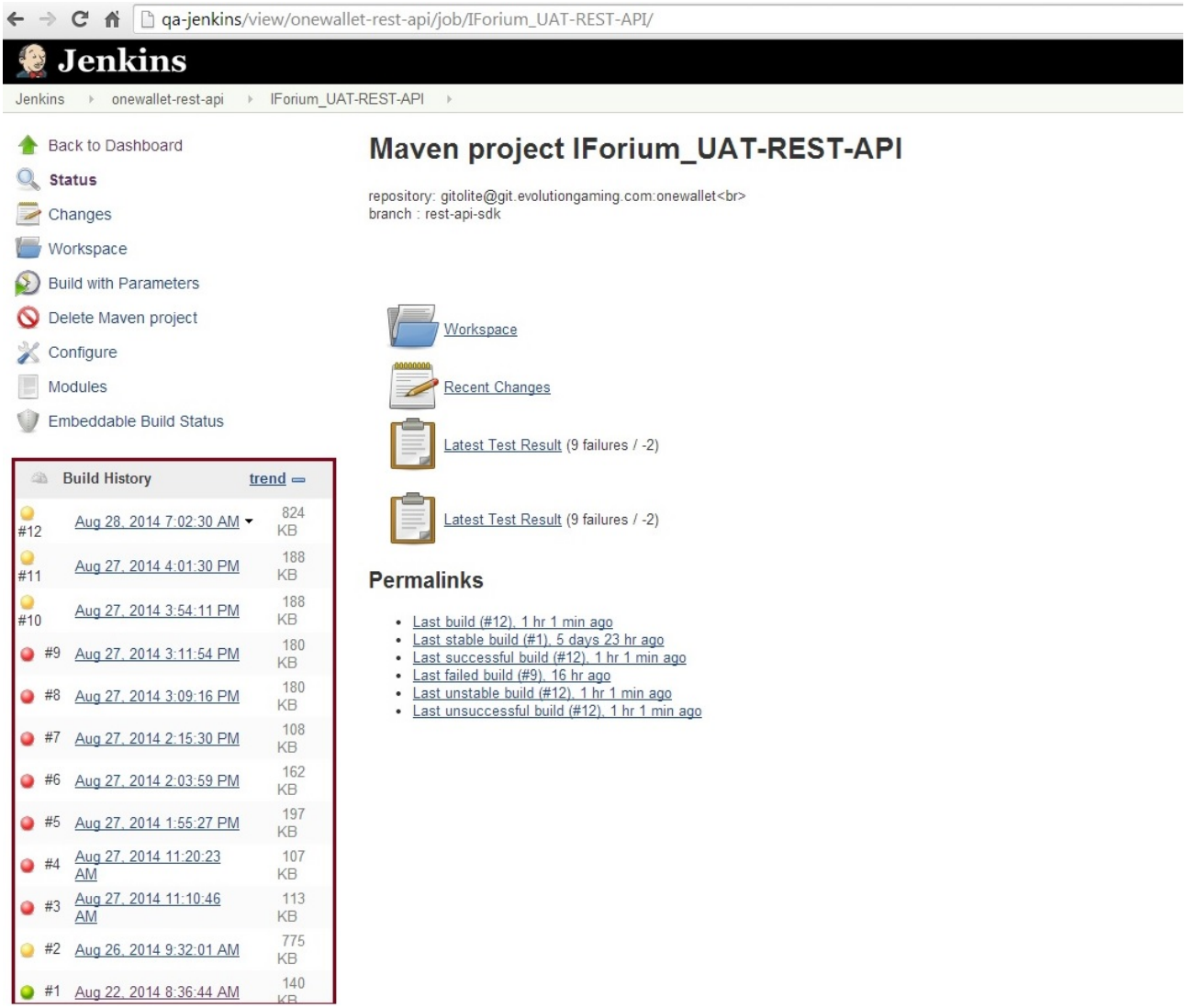
10.1 2. In horizontal menu tab select: onewallet-rest-api



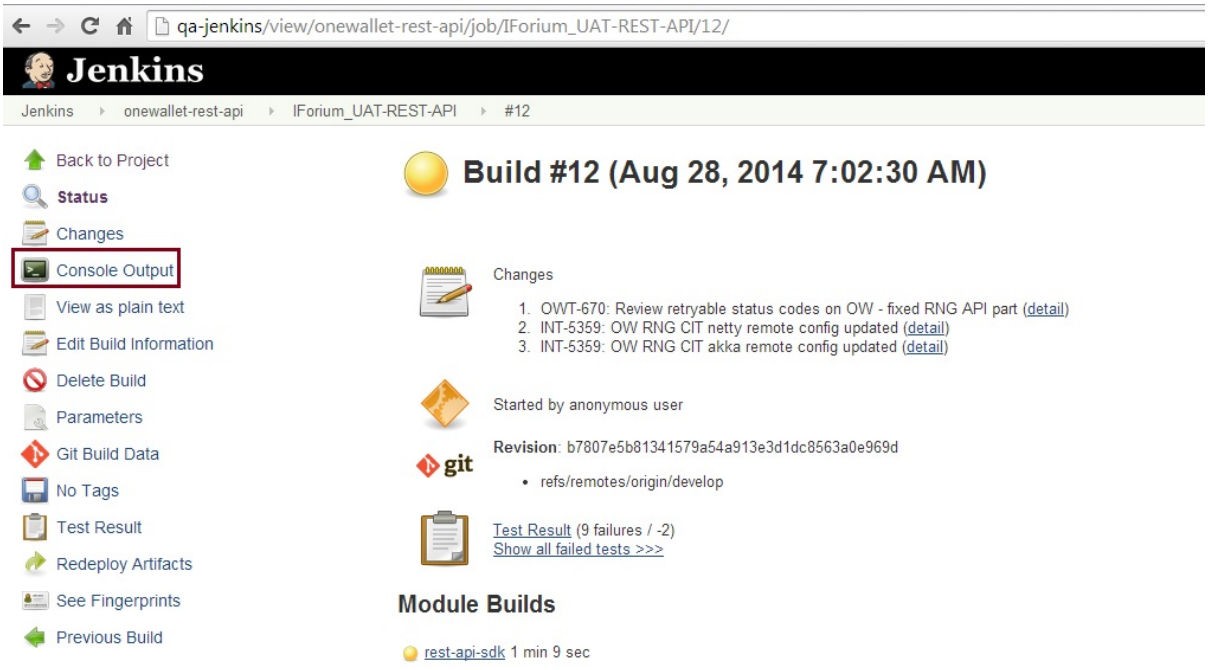
10.2 3. In the vertical list select the desired casino by a click:



10.3 4. In the left window side select one of the builds by a click:



10.4 **5.** In the left menu bar select Console Output:



10.5 **6.** Scroll the page down to "TESTING COMPLETED" where you will find a summary of tests' execution:

- successfully test count (1)
- failure test count (1)
- all executed tests list, including:
    - game type (2),
    - userID (3),
    - execution result: complete successfully OR complete with failures (4).
- failure case list (5).

```
qa-jenkins/view/onewallet-rest-api/job/IForium_UAT-REST-API/12/console
Jenkins › onewallet-rest-api › IForium_UAT-REST-API › #12
TESTING COMPLETED 1 of 1
  "bonus" : null,
  "retransmission" : false
}
2014-08-28 07:04:00,012 [-975382960] [main] ERROR com.evolutiongaming.restapisdk.BaseTest - DEBIT_CREDIT:  failureExpected[OK]: call service debit() failed, status[UNKNOWN_ERROR]
2014-08-28 07:04:00,013 [-975382960] [main] DEBUG com.evolutiongaming.restapisdk.executor.ServiceExecutor - DEBIT_CREDIT: call service debit() done
2014-08-28 07:04:00,014 [-975382960] [main] ERROR com.evolutiongaming.restapisdk.executor.AdapterExecutor - DEBIT_CREDIT: call adapter callWithdrawal() failed
2014-08-28 07:04:00,014 [-975382960] [main] DEBUG com.evolutiongaming.restapisdk.executor.AdapterExecutor - DEBIT_CREDIT: call adapter callWithdrawal() done
2014-08-28 07:04:00,015 [-975382960] [main] ERROR com.evolutiongaming.restapisdk.scenario.SetOfScenarios - DEBIT_CREDIT: debit scenario failed response is null
2014-08-28 07:04:00,016 [-975382960] [main] DEBUG com.evolutiongaming.restapisdk.scenario.SetOfScenarios - DEBIT_CREDIT: debit scenario done
2014-08-28 07:04:00,016 [-975382960] [main] ERROR com.evolutiongaming.restapisdk.scenario.SetOfScenarios - DEBIT_CREDIT: Scenario for game Roulette with failure.
2014-08-28 07:04:00,017 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.scenario.SetOfScenarios - DEBIT_CREDIT: Done scenario for game Roulette
2014-08-28 07:04:00,020 [-975382960] [main] ERROR com.evolutiongaming.restapisdk.BaseTest - TESTING COMPLETED WITH FAILURES
2014-08-28 07:04:00,021 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - SUCCESSFULLY TEST COUNT IS: 2        1
2014-08-28 07:04:00,021 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - FAILURE TEST COUNT IS: 9

2014-08-28 07:04:00,022 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: LOBBY_BALANCE for service [IForium], for game [game null, its lobby balance], userID [T255-
146201] - complete successfully 4
2014-08-28 07:04:00,024 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: LOBBY_BALANCE for service [IForium], for game [game null, its lobby balance], userID [T255-
146202] - complete successfully
2014-08-28 07:04:00,024 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: DEBIT_CREDIT_WITH_DIFFERENT_SIDS for service [IForium], for game [Roulette] 2 userID [T255-
146201] - complete with failures
2014-08-28 07:04:00,025 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: DEBIT_CREDIT_BET_ALREADY_SETTLED for service [IForium], for game [Roulette], userID [T255-
146201] - complete with failures 4
2014-08-28 07:04:00,025 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: DEBIT_CANCEL for service [IForium], for game [Roulette], userID [T255-146201] 3 complete
with failures
2014-08-28 07:04:00,026 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: TWO_USERS_BJ_GAME_EACH_PLAYING_2_HANDS for service [IForium], for game [0000000000000003],
userID [T255-146201&T255-146202] - complete with failures
2014-08-28 07:04:00,026 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: VERIFY_CAN_PLACE_BET_BEFORE_SETTLE_FOR_ANOTHER_GAME for service [IForium], for game
[Roulette], userID [T255-146201] - complete with failures
2014-08-28 07:04:00,026 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: SAME_USER_PLAYING_WITH_DIFFERENT_CHANNEL_TYPE for service [IForium], for game
[Roulette&Blackjack], userID [T255-146201] - complete with failures
2014-08-28 07:04:00,027 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: DEBIT_BET_ALREADY_EXIST for service [IForium], for game [Roulette], userID [T255-146201] -
complete with failures
2014-08-28 07:04:00,027 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: DEBIT_CREDIT_BET_DOES_NOT_EXIST for service [IForium], for game [Roulette], userID [T255-
146201] - complete with failures
2014-08-28 07:04:00,028 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - Scenario: DEBIT_CREDIT for service [IForium], for game [Roulette], userID [T255-146201] - complete
with failures
2014-08-28 07:04:00,028 [-975382960] [main] INFO  com.evolutiongaming.restapisdk.BaseTest - //-----------------------------------------------

Tests run: 11, Failures: 9, Errors: 0, Skipped: 0, Time elapsed: 21.824 sec <<< FAILURE!
executeDebitCreditWithDifferentSIDS[service [IForium]](com.evolutiongaming.restapisdk.EvoStdRestTest)  Time elapsed: 6.098 sec  <<< FAILURE!
java.lang.AssertionError: DEBIT_CREDIT_WITH_DIFFERENT_SIDS for service [IForium], for game [Roulette], userID [T255-146201]
        at org.junit.Assert.fail(Assert.java:88)
        at org.junit.Assert.assertTrue(Assert.java:41)
        at com.evolutiongaming.restapisdk.EvoStdRestTest.executeDebitCreditWithDifferentSIDS(EvoStdRestTest.java:182)

executeDebitCreditBetAlreadySettledScenario[service [IForium]](com.evolutiongaming.restapisdk.EvoStdRestTest)  Time elapsed: 5.224 sec  <<< FAILURE!  5
java.lang.AssertionError: DEBIT_CREDIT_BET_ALREADY_SETTLED for service [IForium], for game [Roulette], userID [T255-146201]
        at org.junit.Assert.fail(Assert.java:88)
        at org.junit.Assert.assertTrue(Assert.java:41)
        at com.evolutiongaming.restapisdk.EvoStdRestTest.executeDebitCreditBetAlreadySettledScenario(EvoStdRestTest.java:153)
```

11 SID for REST API tests

11.1

- What are the ways to get SID value? Predefined SID value Non-predefined SID value What is SID method? Example of how it works with a non-predefined SID value: What are the ways to get SID value?
  - Predefined SID value
  - Non-predefined SID value
- What is SID method?
- Example of how it works with a non-predefined SID value:

What are the ways to get SID value?

SID value for REST API automated tests can be:

- either **predifined**;
- or retrieved by SID method: SID value generated per every method call case (**non-predefined**).

11.1.1 Predefined SID value

A **predefined** SID value is a constant value declared by client and configured respectively. A **predefined** SID value has the following possible events flow:

- SID method is not implemented on customers' side. SID method is not called and Evolution system uses the predefined SID value for every call within REST API automated tests.

11.1.2 Non-predefined SID value

For a non-predefined SID value Evolution will request operator to provide a SID value by calling **SID method**.

There are following possible events flows:

- provided SID value will be re-used for all further call (i.e. initialization, balance, debit);
- after SID value provided subsequent initialization call follows up and a new SID value might be generated (IF the change of SID value within initialization process is accepted by customer).

11.2 What is SID method?

SID method is call/request Evolution will post to operator's platform in order to extract a valid sid value to re-use for further calls (e.g. balance, debit). Assuming that REST service is deployed on URL https://my.service.host.com/api/ and authentication token value ("authToken" parameter) is "s3cr3tV4lu3", sid method call will look in the following way:

https://my.service.host.com/api/sid?authToken=s3cr3tV4lu3

11.3 Example of how it works with a non-predefined SID value:

1. CheckUserRequest

{

"sid":"",
"userId":"euID-parameter-from-UserAuthentication-call",
"channel":{
"type":"P"
},
"uuid":"ce186440-ed92-11e3-ac10-0800200c9a66"
}

2. Where empty "sid" is the case, as the value is not known yet. Request is posted to https://my.service.host.com/api/sid?authToken=s3cr3tV4lu3**;**

3. Operator's side generates a sid value and returns to Evolution in next step[1]**;**

4. CheckUserResponse

{
"status":"OK",
"sid:"new-sid-to-be-used-for-api-calls-qwerty",
"uuid":"ce186440-ed92-11e3-ac10-0800200c9a66"
}

5. SID value received and will be used for further calls as described above.

1 https://my.service.host.com/api/sid?authToken=s3cr3tV4lu3