

**Spadegaming E-Games Common Wallet
Specification Document
(English Version)**

Version 5.2

Revision History

Date	Version	Remarks
2013-11-05	0.1	Convert Chinese version to English
2013-12-12	Version0.2	Edit4.4 transfer, 1: ticketId change Mandatory To NO, 2: Remove referenceId;
2013-12-12	Version0.3	Add 4.7 Query player valid sessionid, tokenid;
2013-12-12	Version0.4	Add 4.8 Query Jackpot pool;
2014-02-24	Version0.5	1. Appendix 4 Game code and new game code 2. ALL param 'acctId' length change from 20 to 50
2014-04-30	Version0.6	Appendix 4 Game code and new game code;
2014-05-05	Version0.7	Add 4.9 Query merchant game info
2014-08-14	Version 0.8	1. Add 4.10 Force Logout for online members 2. Add "cancelId" for Transfer Request 3. Add "sequence" for BetInfo
2014-11-01	Version 0.9	1. Remove 4.7 Query player valid sessionid, tokenid 2. Modify "3 API Interface Overview", add the description for "serialNo" 3. Add description for "4.8 Query merchant game info" 4. Modify "Appendix 2 Language Code" 5. Tidy example code as appendix
2014-12-15	Version 1.0	4.1 Game Lobby Page, add parameter minigame = true/false 4.6 Query transfer history, add parameter Merchant Tx Id
2014-01-26	Version 1.1	4.4 Take away cancel payout and uncanceled payout
2015-03-13	Version 1.2	Appendix 4 Game code and new game code 4.5 GetBetHistory, Result Column extend to varchar (1024)
2015-05-04	Version 1.3	4.1 Game Lobby Page, add parameter mobile= true/false 4.5 GetBetHistory, add one column Channel = Mobile/Web 4.8 GetGames, add one column mthumbnail
2015-11-01	Version 1.4	1. Added 18 New Games 2. 4.14 Query Progressive Jackpot Pool
2015-11-17	Version 1.5	4.11 Install Version User Authorization
2015-12-07	Version 1.6	1.Added 3 New Games
2015-12-17	Version 1.7	1.Added 4 New Games
2016-01-20	Version 1.8	1.Added 16 New Games 2.GetBetHistory, add one column balance
2016-04-10	Version 1.9	4.5 GetBetHistory, add one column jpWin = -100.000000
2016-05-30	Version 2.0	1. 4.4 Transfer, add one column referenceId = afdshfds 2. 4.5 GetBetHistory, add one column referenceId=askdfd 3. Added 11 New Games
2016-10-27	Version 2.1	1. 4.1 Add in Parameter Menumode

		2. 4.12 Add in Parameter JackpotCode & JackpotName 3. Added 11 New Games
2017-01-09	Version 2.2	1. Added 17 New Games
2017-03-01	Version 2.3	1. Added 8 New Games
2017-04-19	Version 2.4	1. Added 2 New Games
2017-08-07	Version 2.5	1. 4.5 getBetHistory - categoryId 'BN' added for angpow function 2. 4.4 Transfer (request)- add Channel (Web, Mobile, APP-i, APP-A,PC) 3. 4.6 Query Transfer history (response) - add Channel (Web, Mobile, APP-i, APP-A,PC) 4. Response code – add 109 Reference No Not found added 5. 4.4 Transfer- Add type 7=Bonus
2017-08-21	2.6	4.1 Game Lobby Page add parameter lobby=SG
2017-12-11	2.7	4.2 User Authorization AcctInfo add in 'siteId'
2018-01-04	2.8	1. Appendix 1 Response Code add '112,113' 2. 4.2 User Authorization - Add note for account id format 3. 4.4 Transfer – Transfer request add in 'specialGame(type, count, sequence)', 'refTicketIds'
2018-08-31	2.9	1. Add 4.12 Create Token API and 4.13 Ticket Details Page 2. Remove 4.12 Install Version User Authorization 3. getGames request param add 'currency'
2019-03-27	3.0	Add date format error code
2019-05-14	3.1	4.3 Add in parameter gameCode
2019-06-18	3.2	1.Add 4.12 createAcct API 2.Add 4.13 Query checkAcct API 3.Add acct exist error code 4.Add format invalid error code 5.Add currency INR and BDT
2019-09-12	3.3	4.6 transferHistory – Add transferId in request
2019-09-17	3.4	4.5 Bet Info Add sequence 4
2019-10-02	3.5	1. Appendix 3 add new currency code ID2 and VN2 2.Update 4.5 BetIp data type to Varchar(50)
2019-12-16	3.6	1. Remove createAcct API 2. Add 4.13 playerDailySumByGame API
2020-01-10	3.7	1. getBetHistory- Control the call limit
2020-02-27	3.8	1. Remove content appendix 4 2. 4.6 Query Transfer history (response) - add in variation of Type
2020-06-10	3.9	1. Add 4.1 exitUrl
2020-09-14	4.0	1. Update 4.5 getBetHistory notes
2020-10-20	4.1	1. Add Russia, Turkey language 2. removed Digest in 1. Protocol

		3. Removed minigame parameter in 4.1 Game Lobby Page
2021-02-16	4.2	1. update explanation of status parameter 2. Add fullScreen parameter in Redirect To Game Lobby
2021-05-10	4.3	1. Add Digest in Header (enhancement) 2. Update currency list Appendix 3 Currency Code Definition
2021-11-25	4.4	1. Optimize the definition of response codes 1. Remove unnecessary error code 2. Add the used error code 2. Protocol adjustment 1. Digest changed to Mandatory 3. Add new currency-AZN
2021-12-10	4.5	1. Transfer split out 2 conditions. 1. 4.4 Transfer and 4.5 Bonus Transfer
2022-04-18	4.6	1. Add new API: 4.9 jackpotPoolAll
2022-06-02	4.7	1. Add new API 4.16 setYoutuber
2022-09-12	4.8	1. getGames - Add-on new param: language 2. Transfer -Add-on new param: gameFeature
2022-10-03	4.9	1. Add new currency code: Appendix 3
2022-11-01	5.0	1. getBetHistory -Add-on new param: gameFeature
2022-12-14	5. 1	1. Add Spanish, French language 2. Updated currency code: Appendix 3 3. remove unnecessary categoryId :BC
2023-02-22	5.2	1. Updated currency code: Appendix 3 2. checkAcct - Add-on new param : includeType, acctType 3. Optimize the definition of response codes - Remove unnecessary error code

Table of content

1. Protocol.....	6
2. Glossary	7
3. API Interface Overview	8
4. API Interface Definition.....	9
4.1 Game Lobby Page.....	9
4.2 User Authorization	11
4.3 Query User Balance	14
4.4 Transfer	17
4.5 Bonus Transfer	23
4.6 Query player betting history	27
4.7 Query transfer history	32
4.8 Query Jackpot pool	35
4.9 Query Jackpot pool of merchant currency	37
4.10 Query merchant game info.....	40
4.11 Force Logout for online members.....	43
4.12 Create Token	45
4.13 Ticket Details Page	47
4.14 Query Exist of player Acct.....	49
4.15 Query player daily summary by game	51
4.16 Set Youtuber	56
Appendix 1 Response Code Definition.....	58
Appendix 2 Language Code Definition	59
Appendix 3 Currency Code Definition (ISO)	60
Appendix 4 API Sample Codes in Java(HTTP).....	62
Appendix 5 API Sample Codes in C#(HTTP)	65
Appendix 6 Authorize Example Code (merchant).....	68
Appendix 7 RSA Decryption Example	69

1. Protocol

HTTP(s) are used to send and receive data. Data of type XML and JSON are supported. Refer to the <http://www.json.org/> for more information of JSON.

In addition to the standard HTTP protocol, the following HTTP Headers are required:

HTTP Header	Mandatory	Example	Description
API	Yes	authorize	Service Name
DataType	Yes	JSON	Data Type (XML or JSON)
Digest	Yes	MD5(Data+securityKey)	Message digest of the post data -securityKey will provide by API Provider
Accept-Encoding	No	gzip, deflate	Identify whether or not support gzip compression of the returning data
Content-Encoding	No	gzip	Identify whether or not using gzip compression of the post data
Accept-Language	No	en_US	Set the default language of the returning data.

Note:

Please use UTF-8 as the default encoding for all encode/decode codes when convert text to/from byte[], without any spaces and line breaks .

```
public abstract class DigestUtils
{
    public static String digest(byte[] input) {
        byte[] digest = Digest.md5().compute( input );
        return Hex.encodeHexString( digest );
    }

    public static String digest(byte[] input, byte[]secretKey) {
        int bodySize=input.length,keySize=secretKey.length;
        byte[] buffer= new byte[bodySize+keySize];
        System.arraycopy(input,0,buffer,0,bodySize);
        System.arraycopy(secretKey,0,buffer,bodySize,keySize);
        byte[] digest = Digest.md5().compute( input );
        return Hex.encodeHexString(digest);
    }
}
```

2. Glossary

1. Game Provider: e-games development team.
2. Merchant: The party that wants to integrate e-games to their website.
3. API Requestor: The party responsible for calling the function
4. API provider: The party responsible for hosting the function
5. Acct ID: Player's unique identity code

3. API Interface Overview

During API call, the invoker will create the request message according to API specification, and post the data to the URL. API provider will send the response message after processing the request.

Usually the message will be defined as: `xxxResponse xxxx(xxxRequest request)`

Where *xxxRequest* equates to API request message and *xxxResponse* equates to API response message.

All request messages will contain the following fields:

Name	Data Type	Example	Description
serialNo	Varchar(50)	20120722224255982841	Generate by merchant to indicate message sequence.
merchantCode	Varchar(10)	SPADE	Unique code that identifies the Merchant.

All response messages will contains the following fields:

Name	Data Type	Example	Description
serialNo	Varchar(50)	20120722224255982841	Generated by merchant to indicate message sequence.
code	Integer	0	Processed result status.
msg	Varchar(128)	success	Processed result description.
merchantCode	Varchar(10)	SPADE	Unique code that identifies the Merchant.

Please refer to appendix 1 for the list of status codes that system returns.

Warning: serialNo is the identity of the message, and one serialNo for one request or response. Please make sure the serialNo is unique.

We advise use GUID (UUID), or the date+time+random string as the serialNo.

4. API Interface Definition

4.1 Game Lobby Page

Usage: Player enters E-games page by clicking *Goto Game* or *Real Money* button found in merchant website.

Interface Name: **Redirect To Game Lobby**

API Provider: Game Provider

API Requestor: Merchant

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
token	Varchar(80)	Yes	fe1a85adc54545d29	Generate by merchant side
language	Varchar(10)	No	en_US	Language to display in game lobby
game	Varchar(10)	Yes	S-DG02	game code
fun	Varchar(10)	No	true	play for fun or real money
mobile	Varchar(10)	No	true/false	Play in mobile
menumode	Varchar(10)	No	On/off	Game Menu Switch off
exitUrl	Varchar(150)	No	http://www.baidu.com	Redirect Url to exit button
fullScreen	Varchar(10)	No	on/off	Full screen Switch off

Note:

1. For language, Refer to appendix 2.
2. If play for fun, the URL must contains acctId and token, fun=true
3. If play for mobile game, must contains mobile=true
4. If not required game menu, must contains menumode=off
5. If not required full screen, must contains fullScreen=off
6. If needs to login game lobby, must replace game=S-XXXX to lobby=SG
7. This function will be not for use to calling it, just introduce the parameter in game url

Game Lobby Redirect URL and Parameters (Real money):

`http://portal.e-games.com/auth/?acctId=TESTPLAYER1&language=en_US&
token=fe1a85adc54545d2963b661a22d09c9e&game=S-DG02&mobile=true
&menumode=on&fullScreen=on&exitUrl=http://www.baidu.com`

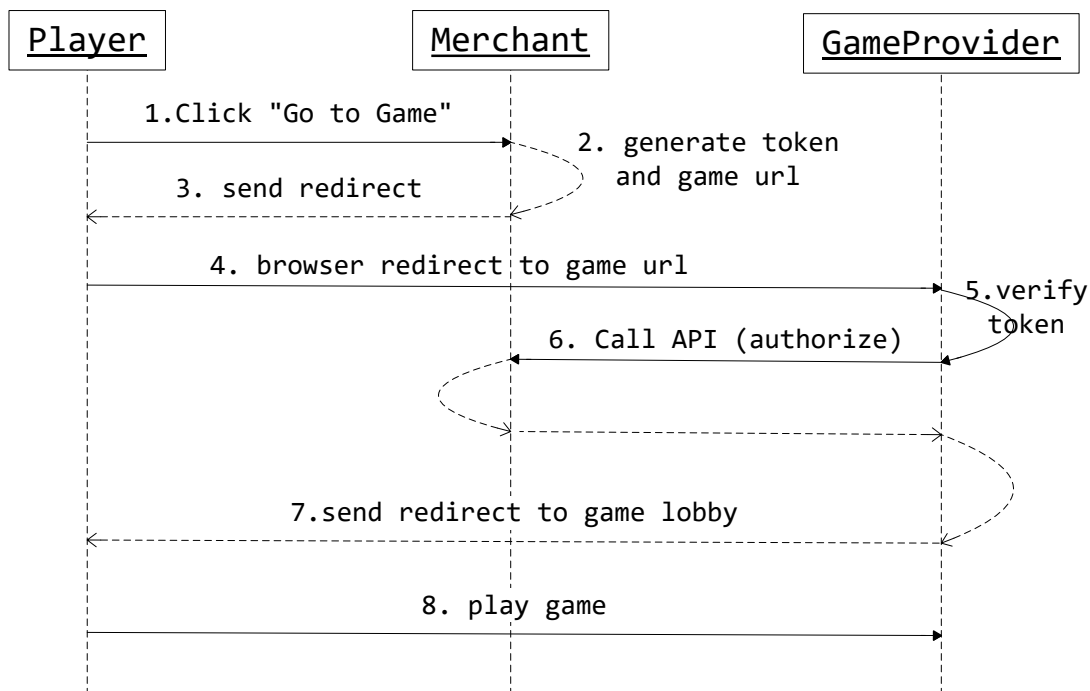
Game Lobby Redirect URL and Parameters (play for fun)

`http://portal.e-games.com/auth/?acctId=TESTPLAYER1&language=en_US&
token=fe1a85adc54545d2963b661a22d09c9e&game=S-DG02&fun=true`

&mobile=true&menumode=on&exitUrl=http://www.baidu.com

Game Lobby Redirect URL and Parameters(Lobby):

http://portal.e-games.com/auth/?acctId=TESTPLAYER1&language=en_US&
token=fe1a85adc54545d2963b661a22d09c9e&lobby=SG&mobile=true



4.2 User Authorization

Usage: Merchant will provide this request for Game provider to authenticate the session.

Interface Name: **authorize**

API Provider: Merchant

API Requestor: Game Provider

Request Message Definition for AuthorizeRequest:

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
token	Varchar(80)	Yes	fe1a85adc54545d2963b661a22d09c9e	Generate by merchant side
language	Varchar(10)	No	en_US	Language code.
serialNo	Varchar(50)	Yes	20120722224255982841	generated by "requestor"
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code

Response message definition for AuthorizeResponse:

Name	Data Type	Mandatory	Example
acctInfo	AcctInfo		AcctInfo Type

AcctInfo type definition:

Name	Data Type	Mandatory	Example	Name
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
userName	Varchar(15)	No	TESTPlayer1	Player name
currency	Char(3)	Yes	USD	Currency Code
balance	Decimal(20,4)	Yes	1000	Current player balance
siteId	Varchar (10)	No	SITE_USD1	Unique site ID for merchant

Note:

1. Refer to appendix 3 for a list of currency code that system support.
2. Acct ID format: [a-zA-Z0-9_-@]{1,50}. Example: TestPlayer_1

Authorize (Request and response messages example)

JSON AuthorizeRequest:

```
{
  "acctId": "TESTPLAYER1",
  "language": "en_US",
  "merchantCode": "SPADE",
  "token": "fe1a85adc54545d2963b661a22d09c9e",
  "serialNo": "20120722224255982841"
}
```

XML AuthorizeRequest:

```
<AuthorizeRequest>
  <serialNo>20120723161423158480</serialNo>
  <token>fe1a85adc54545d2963b661a22d09c9e</token>
  <merchantCode>SPADE</merchantCode>
  <acctId>TESTPLAYER1</acctId>
  <language>en_US</language>
</AuthorizeRequest>
```

JSON AuthorizeResponse:

```
{
  "acctInfo": {
    "acctId": "TESTPLAYER1",
    "balance": 0,
    "userName": "TESTPlayer1",
    "currency": "USD",
    "siteId": "SITE_USD"
  },
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120722224255982841"
}
```

XML AuthorizeResponse:

```
<AuthorizeResponse>
  <serialNo>20120722224255982841</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <acctInfo>
    <acctId>TESTPLAYER1</acctId>
    <userName>TESTPlayer1</userName>
    <currency>USD</currency>
    <balance>0</balance>
    <siteId>SITE_USD</siteId>
  </acctInfo>
</AuthorizeResponse>
```

4.3 Query User Balance

Usage scenario: Search for user balance .

Interface Name: **getBalance**

API Provider: Merchant

API Requestor: Game Provider

Request Message Definition for **GetBalanceRequest**:

Name	Data Type	Mandatory	Example	Remark
acctId	Varchar(50)	Yes	TESTPLAYER1	Acct Id
gameCode	Varchar(10)	No	S-DG02	Game Code

Response message definition for **GetBalanceResponse**:

Name	Data Type	Mandatory	Example	Remark
acctInfo	AcctInfo	Yes	AcctInfo	acctInfo

AcctInfo type definition:

Name	Data Type	Mandatory	Example	Name
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
userName	Varchar(15)	No	TESTPlayer1	Player name
currency	Char(3)	Yes	USD	Currency Code
balance	Decimal(20,4)	Yes	1000	Current player balance
siteId	Varchar (10)	No	SITE_USD1	Unique site ID for merchant

GetBalance Request and response messages example :

JSON GetBalanceRequest:

```
{
  "acctId": "TESTPLAYER1"
  "merchantCode": "SPADE",
  "serialNo": "20120722224255982841"
  "gameCode": "S-DG02"
}
```

XML GetBalanceRequest:

```
<GetBalanceRequest>
  <serialNo>20120722224255982841</serialNo>
  <merchantCode>SPADE</merchantCode>
  <acctId>TESTPLAYER1</acctId>
  <gameCode>S-DG02</gameCode>
</GetBalanceRequest>
```

JSON GetBalanceResponse:

```
{
  "acctInfo": {
    "userName": "TestPlayer1",
    "currency": "USD",
    "acctId": "TESTPLAYER1",
    "balance": 1000
  },
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120722224255982841"
}
```

XML GetBalanceResponse:

```
<GetBalanceResponse>
  <serialNo>20120722224255982841</serialNo>
  <code>0</code>
  <msg>0</msg>
  <merchantCode>SPADE</merchantCode>
  <acctInfo>
    <acctId>TESTPLAYER1</acctId>
    <userName>TestPlayer1</userName>
    <currency>USD</currency>
    <balance>1000.0000</balance>
  </acctInfo>
</GetBalanceResponse>
```


4.4 Transfer

Usage:

- 1) When player place bet, Game Provider will send the transfer request to merchant, and merchant will deduct the player balance.
- 2) When game provider payout, Game Provider will send the transfer request to merchant, and merchant will increase the player balance.

Interface Name: **transfer**

API Provider: Merchant

API Requestor: Game Provider

Request Message Definition for **TransferRequest**:

Name	Data Type	Mandatory	Example	Description
transferId	Varchar(50)	Yes	a3b0c9dd1ab041	Unique transfer Id
acctId	Varchar(50)	Yes	TESTPLAYER1	Player id
currency	Char(3)	Yes	USD	Currency code
amount	Decimal(20,9)	Yes	1000	Transfer amount (greater than or equal to 0)
type	Int	Yes	1	1 = place bet 2 = cancel bet 4 = payout
channel	Varchar(10)	Yes	Web/Mobile, APP-i/APP-A/PC	Ticket for mobile or web
gameCode	Varchar(10)	Yes	S-DG02	Game Code
ticketId	Varchar(20)	No	641482277	
referenceId	Varchar(50)	No	a8ab9cc8f1a277de	Reference to place bet transfer id
specialGame	SpecialGame		SpecialGame Type	Only get bonus, free, bonusfree, freebonus will send this
refTicketIds	Varchar(2048)	No	[120001, 120002, 120003]	Reference ticket ids for cascade game.
gameFeature	Varchar(50)	No	BUY-8	Only available when player trigger "Buy Feature". suggestion: judge the prefix BUY, Whenever a

				gameFeature contains the BUY prefix, it is a "Buy Feature"
--	--	--	--	--

SpecialGame type definition:

Name	Data Type	Mandatory	Example	Description
type	Varchar(20)	No	Free	Differentiate the special game type. Example: bonus, free, bonusfree, freebonus
count	Int	No	10	Total count on the special game
sequence	Int	No	1	Count on the completed special game

specialGame & refTicketIds only will pass during type = 4, payout

Important Note:

Type 4: Payout

1. If the transfer Id received is duplicated, merchant need to check the transfer status and reply the status to game provider. Merchant CAN NOT process the transfer with same transfer id, just respond us success and take no action to player balance

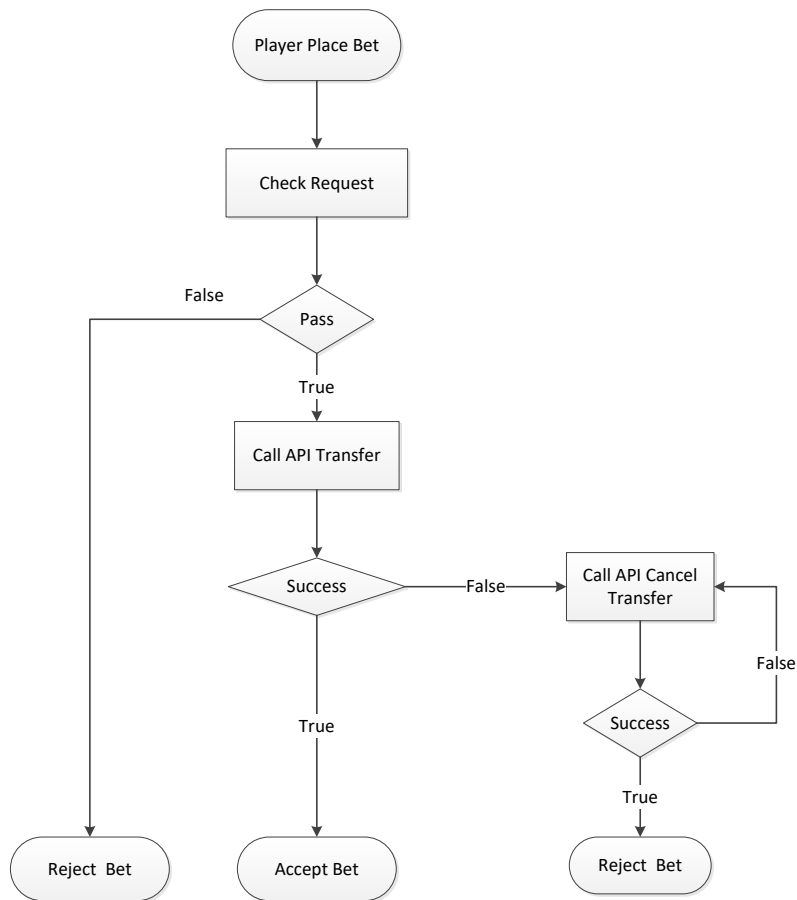
Type 2: Cancel Bet

1. If the reference Id are duplicated or no found on the cancel bet (type=2), Merchant should return us Error Code: please response us code 0 or 109. Except this error code, provider will continue resend you the reference id.
2. If placeBet encounter timeout (our system did not receive any response), cancel bet will be sent.
3. Even with successful placeBet, under certain circumstances also able to trigger cancel bet. This will happen when the player enters the fishing game and leave the game without any betting (shooting).

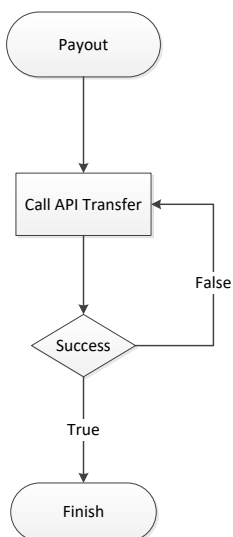
Response message definition for [TransferResponse](#):

Name	Data Type	Mandatory	Example	Description
transferId	Varchar(50)	Yes	a3b0c9dd1ab041	Unique transfer Id
merchantTxId	Varchar(20)	No	498036704	Merchant transaction id
acctId	Varchar(50)	Yes	TESTPLAYER1	Player ID
balance	Decimal(20,4)	Yes	1000	Player balance after transfer

Place Bet Flow:



Payout Flow:



Transfer Request and response messages example:

JSON TransferRequest:

```
{
  "acctId": "TESTPLAYER1",
  "transferId": "0ab9bdca06c14811b24653468e60988",
  "currency": "USD",
  "amount": 10,
  "type": 4,
  "ticketId": "234950357",
  "channel": "Web",
  "gameCode": "S-DG02",
  "merchantCode": "SPADE",
  "serialNo": "20120722231413699735",
  "referenceId": "dffffdca06c14811b24653468e60",
  "specialGame": {
    "type": "Free",
    "count": 10,
    "sequence": 1
  },
  "refTicketIds": [
    "120001, 120002, 120003"
  ]
}
```

XML TransferRequest:

```
<TransferRequest>
  <serialNo>20120722231413699735</serialNo>
  <merchantCode>SPADE</merchantCode>
  <transferId>0ab9bdca06c14811b24653468e60988</transferId>
  <acctId>TESTPLAYER1</acctId>
  <currency>USD</currency>
  <amount>10</amount>
  <type>4</type>
  <ticketId>234950357</ticketId>
  <channel>Web</channel>
  <gameCode>S-DG02</gameCode>
  <referenceId>dffffdca06c14811b24653468e60</referenceId>
  <specialGame>
    <type>Free</type>
    <count>10</count>
    <sequence>1</sequence>
  </specialGame>
  <ref_ticket_ids>[120001, 120002, 120003]</ref_ticket_ids>
</TransferRequest>
```

JSON TransferResponse:

```
{
  "transferId": "0ab9bdca06c14811b24653468e609838",
  "merchantCode": "SPADE",
  "merchantTxId": "20130813014319279367",
  "acctId": "Test10001",
  "balance": 1050,
  "msg": "success",
  "code": 0,
  "serialNo": "20120722231413699735"
}
```

XML TransferResponse:

```
<TransferResponse>
  <serialNo>20120722231413699735</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <transferId>0ab9bdca06c14811b24653468e60988</transferId>
  <merchantTxId>20130813014319279367</merchantTxId>
  <acctId>Test10001</acctId>
  <balance>1050</balance>
</TransferResponse>
```

4.5 Bonus Transfer

Usage:

1. Apply on promotion activities. During promotion activities, if players triggered bonus, system would send type= 7 to Merchant.
2. When Game Provider send bonus transfer request (type=7) to merchant, merchant must able to respond and increase the player balance.
3. Merchant must develop this API to participate planned promotion activities in near future.

Interface Name: **transfer**

API Provider: Merchant

API Requestor: Game Provider

Request Message Definition for **TransferRequest**:

Name	Data Type	Mandatory	Example	Description
transferId	Varchar(50)	Yes	a3b0c9dd1ab041	Unique transfer Id
acctId	Varchar(50)	Yes	TESTPLAYER1	Player id
currency	Char(3)	Yes	USD	Currency code
amount	Decimal(20,9)	Yes	1000	Transfer amount (greater than or equal to 0)
type	Int	Yes	7	7 = Bonus
channel	Varchar(10)	Yes	Web/Mobile, APP-i/APP-A/PC	Ticket for mobile or web
gameCode	Varchar(10)	Yes	B-FS02	Game Code
ticketId	Varchar(20)	Yes	641482277	
referenceId	Varchar(50)	No		Don't have reference Id due to don't have placebo
roundId	Varchar(20)	No	432	Promotion's Id
siteId	Varchar(20)	No	DEMO1	Client's site Id

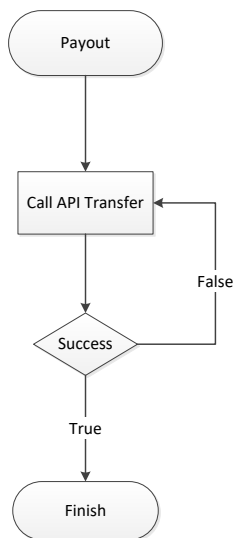
Important Note:

1. Bonus don't have reference Id, due to don't placebet.
2. RoundId and siteId will be enabled upon request

Response message definition for **TransferResponse**:

Name	Data Type	Mandatory	Example	Description
transferId	Varchar(50)	Yes	a3b0c9dd1ab041	Unique transfer Id
merchantTxId	Varchar(20)	No	498036704	Merchant transaction id
acctId	Varchar(50)	Yes	TESTPLAYER1	Player ID
balance	Decimal(20,4)	Yes	1000	Player balance after transfer

Bonus Flow:



Transfer Request and response messages example:

JSON TransferRequest:

```
{
  "acctId": "TESTPLAYER1",
  "transferId": "a3b0c9dd1ab041",
  "currency": "USD",
  "amount": 10,
  "type": 7,
  "ticketId": "234950357",
  "channel": "Web",
  "gameCode": "B-FS02",
  "merchantCode": "SPADE",
  "serialNo": "20120722231413699735",
  "referenceId": ""
}
```

XML TransferRequest:

```
<TransferRequest>
  <serialNo>20120722231413699735</serialNo>
  <merchantCode>SPADE</merchantCode>
  <transferId>a3b0c9dd1ab041</transferId>
  <acctId>TESTPLAYER1</acctId>
  <currency>USD</currency>
  <amount>10</amount>
  <type>7</type>
  <ticketId>234950357</ticketId>
  <channel>Web</channel>
  <gameCode>B-FS02</gameCode>
  <referenceId />
</TransferRequest>
```

JSON TransferResponse:

```
{
  "transferId": "a3b0c9dd1ab041",
  "merchantCode": "SPADE",
  "merchantTxId": "20130813014319279367",
  "acctId": "TESTPLAYER1",
  "balance": 1050,
  "msg": "success",
  "code": 0,
  "serialNo": "20120722231413699735"
}
```

XML TransferResponse:

```
<TransferResponse>
  <serialNo>20120722231413699735</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <transferId>a3b0c9dd1ab041</transferId>
  <merchantTxId>20130813014319279367</merchantTxId>
  <acctId>TESTPLAYER1</acctId>
  <balance>1050</balance>
</TransferResponse>
```

4.6 Query player betting history

Usage: Search for bet history

Interface Name: **getBetHistory**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **GetBetHistoryRequest**:

Name	Data Type	Mandatory	Example	Description
beginDate	Char(15)	Yes	20120720T230043	Start date range to search. Format must be in yyyymmdd'T'24hhmmss
endDate	Char(15)	Yes	20120722T225043	End date range to search. Format must be in yyyymmdd'T'24hhmmss
pageIndex	Integer	Yes	1	Page number
serialNo	Varchar(50)	Yes	20120722231413595602	Generated by merchant to indicate message sequence.
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code

Response message definition for **GetBetHistoryResponse**:

Name	Data Type	Mandatory	Example	Description
resultCount	Integer	Yes	100	Total record count
pageCount	Integer	Yes	1	Total page count
list	BetInfo[]	Yes		BetInfo Array

BetInfo definition:

Name	Data Type	Mandatory	Example	Description
ticketId	Varchar(20)	Yes	641482277	Ticket number.
acctId	Varchar(50)	Yes	TESTPLAYER1	Player's unique Id
ticketTime	Char(15)	Yes	20120722T225417	Date time when ticket is bet
categoryId	Varchar(10)	Yes	SM or TB or AD or BN or FH	Game category
gameCode	Varchar(10)	Yes	S-DG02	Game Code
currency	Char(3)	Yes	USD	Currency ISO code
betAmount	Decimal(18,6)	Yes	20	Bet Amount
result	Varchar(1024)	No	8,8,5,10,10,7,7	Result
winLoss	Decimal(18,6)	Yes	-20	Member win loss.
jackpotAmount	Decimal(18,6)	Yes	0	
betIp	Varchar(50)	Yes	8.8.8.8	Bettor's IP Address
luckyDrawId	Long	No	10	
roundId	Integer	Yes	1823	Game Round ID
sequence	Integer	Yes	0,1,2,3,4	0 = No jackpot win
channel	Varchar(10)	Yes	Mobile/Web	Ticket for mobile or web
Balance	Decimal(18,6)	Yes	234.000000	Previous Balance
jpWin	Decimal(18,6)	Yes	-100.000000	Jackpot Win
referenceId	Varchar(50)	No	a8ab9cc8f1a277de	Reference to transfer id
gameFeature	Varchar(50)	No	BUY-20	Only response when player trigger Buy Feature

Notes :

Ticket is based on game payout datetime. Recommend retrieve interval is within 1-5 minutes and each ticket retrieve after 3 minutes of payout time. Maximum retrieve interval is 2 hours.

It has concurrent call limit for this query, each query will only enable SINGLE request. The next request will only enable after current request has completed. Otherwise, it will return code: 101 USER_CALL_LIMITED

GetBetHistory (Request and response messages example):

JSON GetBetHistoryRequest:

```
{  
  "beginDate": "20120721T175139",  
  "endDate": "20120723T174139",  
  "pageIndex": 1,  
  "merchantCode": "SPADE",  
  "serialNo": "20120723175139440637"  
}
```

XML GetBetHistoryRequest:

```
<GetBetHistoryRequest>  
  <serialNo>20120723175139440637</serialNo>  
  <merchantCode>SPADE</merchantCode>  
  <beginDate>20120721T175139</beginDate>  
  <endDate>20120723T174139</endDate>  
  <pageIndex>1</pageIndex>  
</GetBetHistoryRequest>
```

JSON GetBetHistoryResponse:

```
{
  "resultCount": 1,
  "list": [
    {
      "ticketId": 633161396,
      "acctId": "TESTPLAYER1",
      "categoryId": "SM",
      "gameCode": "SS-GD02",
      "ticketTime": "20130908T161044",
      "betIp": "8.8.8.8",
      "betAmount": 20,
      "winLoss": 30,
      "result": "23412",
      "jackpotAmount": 1.5,
      "luckyDrawId": 0,
      "completed": "True",
      "roundId": 1879,
      "sequence": 0,
      "channel": "web",
      "balance": 234.0,
      "jpWin": 0.0,
      "referenceId": "arwwew232ghfg4dfsdfsdw2",
      "gameFeature": "BUY-20"
    }
  ],
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120723175139440637"
}
```

XML GetBetHistoryResponse:

```
<GetBetHistoryResponse>
  <serialNo>20120723175139440637</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <resultCount>1</resultCount>
  <list>
    <BetInfo>
      <ticketId>633161396</ticketId>
      <acctId>Test10001</acctId>
      <categoryId>SM</categoryId>
      <gameCode>SS-GD02</gameCode>
      <ticketTime>20130908T161044</ticketTime>
      <betIp>8.8.8.8</betIp>
      <betAmount>20</betAmount>
      <winLoss>30</winLoss>
      <jackpotAmount>1.5</jackpotAmount>
      <result>23412</result>
      <luckyDrawId>0</luckyDrawId>
      <completed>True</completed>
      <sequence>0</sequence>
      <roundId>1387</roundId>
      <channel>Web</channel>
      <balance>234.000000</balance>
      <jpWin>0.0</jpWin>
      <referenceId>arwwwew232ghfg4dfsdwsdwr2</referenceId>
      <gameFeature>BUY-20</gameFeature>
    </BetInfo>
  </list>
</GetBetHistoryResponse>
```

4.7 Query transfer history

Usage: Search for transfer history

Interface Name: **transferHistory**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition **TransferHistoryRequest:**

Name	Data Type	Mandatory	Example	Description
beginDate	Char(15)	Yes	20120720T230043	Start date range to search. Format yyyyymmdd'T'24hhmmss
endDate	Char(15)	Yes	20120722T225043	End date range to search. Format yyyyymmdd'T'24hhmmss
pageIndex	Integer	Yes	1	Page number
transferId	Varchar(50)	No	a3b0c9dd1ab041	Unique transfer Id

Response message definition **TransferHistoryResponse:**

Name	Data Type	Mandatory	Example	Description
resultCount	Integer	Yes	100	Total record count
pageCount	Integer	Yes	1	Total page count
list	TransferInfo[]	Yes		TransferInfo Array

TransferInfo Definition **TransferInfo:**

Name	Data Type	Mandatory	Example	Description
transferId	Varchar(50)	Yes	a3b0c9dd1ab041	Unique transfer Id
acctId	Varchar(50)	Yes	11	Player ID
ticketId	Varchar(20)	Yes	641482277	
transferTime	Char(15)	Yes	20120722T225043	Transfer Time
merchantTxid	Varchar(20)	Yes	D141215030051fmowg	Merchant Transfer Id
type	Int	Yes	1	1 = place bet 2 = cancel bet 4 = payout 7 = Bonus
amount	Decimal(20,4)	Yes	1000	Transfer Amount
status	Int	Yes	0	Transfer status
channel	Varchar(10)	Yes	Web/Mobile, APP-i/APP-A/PC	Ticket for mobile or web

TransferHistory Request and response messages example:

JSON TransferHistoryRequest:

```
{
  "beginDate": "20120721T175139",
  "endDate": "20120723T174139",
  "pageIndex": 1,
  "merchantCode": "SPADE",
  "serialNo": "20120723175139440637",
  "transferId": "667706462"
}
```

XML TransferHistoryRequest:

```
<TransferHistoryRequest>
  <serialNo>20120723175139440637</serialNo>
  <merchantCode>SPADE</merchantCode>
  <beginDate>20120721T175139</beginDate>
  <endDate>20120723T174139</endDate>
  <pageIndex>1</pageIndex>
  <transferId>667706462</transferId>
</TransferHistoryRequest>
```

JSON TransferHistoryResponse:

```
{
  "list": [
    {
      "transferId": "667706462",
      "acctId": "Test10001",
      "ticketId": "749084635",
      "type": 1,
      "amount": 20,
      "transferTime": "20130908T151100",
      "merchantTxId": "D141215030051fmowg",
      "status": 0,
      "channel": "Web"
    }
  ],
  "resultCount": 1,
  "pageCount": 1,
  "merchantCode": "SPADE",
  "code": 0,
}
```

```
"msg": "success",  
"serialNo": "20120726214956563472"  
}
```

XML TransferHistoryResponse:

```
<TransferHistoryResponse>  
  <serialNo>20120726214956563472</serialNo>  
  <code>0</code>  
  <msg>success</msg>  
  <merchantCode>SPADE</merchantCode>  
  <resultCount>1</resultCount>  
  <pageCount>1</pageCount>  
  <list>  
    <TransferInfo>  
      <transferId>667706462</transferId>  
      <acctId>Test10001</acctId>  
      <ticketId>749084635</ticketId>  
      <type>1</type>  
      <amount>20</amount>  
      <transferTime>20130908T151100</transferTime>  
  
    <merchantTxId>D141215030051fmowg</merchantTxId>  
      <status>0</status>  
      <channel>Web</channel>  
    </TransferInfo>  
  </list>  
</TransferHistoryResponse>
```

4.8 Query Jackpot pool

Usage: Merchant query the jackpot pool amount and display on merchant website

Interface Name: **jackpotPool**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **JackpotPoolRequest**:

Name	Data Type	Mandatory	Example	Description
currency	Char(3)	Yes	USD	Currency ISO code

** If currency is empty, then all jackpot for each currency will listed

Response message definition for **JackpotPoolResponse**:

Name	Data Type	Mandatory	Example	Description
list	JackpotPoolInfo[]	Yes		JackpotPoolInfo type

JackpotPoolInfo type definition:

Name	Data Type	Mandatory	Example	Description
code	Varchar(20)	Yes	Grand	Jackpot Code
name	Varchar(50)	否	Grand Jackpot	Jackpot Name
amount	Decimal(20,2)	是	3844.08	Jackpot Pool amount

JackpotPool (Request and response messages example):

JSON JackpotPoolRequest:

```
{
  "currency": "USD",
  "merchantCode": "SPADE",
  "serialNo": "20120726214956563472"
}
```

XML JackpotPoolRequest:

```
<JackpotPoolRequest>
  <serialNo>20120726214956563472</serialNo>
  <merchantCode>SPADE</merchantCode>
  <currency>USD</currency>
</JackpotPoolRequest>
```

JSON JackpotPoolResponse:

```
{
  "list": [
    {
      "code": "Grand",
      "name": "Grand Jackpot",
      "amount": 3844.08,
      "currency": "USD"
    }
  ],
  "status": 0,
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120726214956563472"
}
```

XML JackpotPoolResponse:

```
<JackpotPoolResponse>
  <serialNo>20120802152140143938</serialNo>
  <code>0</code>
  <msg>0</msg>
  <merchantCode>SPADE</merchantCode>
  <resultCount>1</resultCount>
  <pageCount>1</pageCount>
  <list>
    <JackpotPoolInfo>
      <code>Grand</code>
      <name>Grand Jackpot</name>
      <amount>3844.08</amount>
      <currency>USD</currency>
    </JackpotPoolInfo>
  </list>
</JackpotPoolResponse>
```

4.9 Query Jackpot pool of merchant currency

Usage: Merchant query the jackpot pool amount and display on merchant website

Interface Name: **jackpotPoolAll**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **JackpotPoolRequest**:

Name	Data Type	Mandatory	Example	Description
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code
serialNo	Varchar(50)	Yes	20120722231413595602	Generated by merchant to indicate message sequence.

Response message definition for **JackpotPoolResponse**:

Name	Data Type	Mandatory	Example	Description
list		Yes		JackpotPoolInfo type

JackpotPoolInfo type definition:

Name	Data Type	Mandatory	Example	Description
code	Varchar(20)	Yes	Grand	Jackpot Code
amoutMap	Varchar(50)	Yes	"CNY":103418.588499	jackpotAmount

JackpotPool (Request and response messages example):

JSON JackpotPoolAllRequest:

```
{
  "merchantCode": "SPADE",
  "serialNo": "20120726214956563472"
}
```

XML JackpotPoolAllRequest:

```
<JackpotPoolAllRequest>
  <serialNo>20120726214956563472</serialNo>
  <merchantCode>SPADE</merchantCode>
</JackpotPoolAllRequest>
```

JSON JackpotPoolAllResponse:

```
{
  "list": [
    {
      "code": "Grand",
      "amountMap": {
        "CNY": 103418.588499,
        "IDR": 3991172.548579,
        "HKD": 2175666.803645
      }
    },
    {
      "code": "GJ",
      "amountMap": {
        "CNY": 16494.54591,
        "IDR": 30333469.927946,
        "HKD": 1944453.200509
      }
    }
  ]
  "status": 0,
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120726214956563472"
}
```

XML JackpotPoolResponse:

```
<JackpotPoolAllResponse>
  <serialNo>20120802152140143938</serialNo>
  <code>0</code>
  <msg>0</msg>
  <merchantCode>SPADE</merchantCode>
  <resultCount>1</resultCount>
  <pageCount>1</pageCount>
  <list>
    <code>Grand</code>
    <amountMap>
      <CNY>103418.588499</CNY>
      <IDR>3991172.548579</IDR>
      <HKD>2175666.803645</HKD>
    </amountMap>
    <code>GJ</code>
    <amountMap>
      <CNY>16494.54591</CNY>
      <IDR>30333469.927946</IDR>
      <HKD>1944453.200509</HKD>
    </amountMap>
  </list>
</JackpotPoolAllResponse>
```

4.10 Query merchant game info

Usage: Query merchant game info

Interface Name: **getGames**

API Provider: Game Provider

API Requestor: Merchant

Request message definition for **MerchantGamesRequest**:

Name	Data Type	Mandatory	Example	Description
serialNo	Varchar(50)	Yes	20120722231413595602	Generated by merchant to indicate message sequence.
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code
currency	Char(3)	No	USD	Currency Code
language	Varchar(10)	No	th_TH (Appendix 2)	Language code

Response message definition for **MerchantGamesResponse**:

Name	Data Type	Mandatory	Example	Description
games	GameInfo[]	yes		Game Info class

GameInfo Definition **GameInfo**:

Name	Data Type	Mandatory	Example	Description
gameCode	Varchar(10)	yes	SS-GD02	Game code
gameName	Varchar(30)	yes	DerbyNight	Game name
jackpot	boolean	yes	True	Is jackpot
thumbnail	Varchar(100)	yes	/images/aa.jpg	Game picture
screenshot	Varchar(100)	yes	/images/bb.jpg	Game screen shot
jackpotCode	Varchar(50)	no	Holy	Only for Jackpot Game
jackpotName	Varchar(50)	no	Holy Jackpot	Only for Jackpot Game

Note:

- The full path of the picture is `http://$host/thumbnail/aa.jpg`, `http://$host/screenshot/bb.jpg`. **\$host** is the host part of the API URL.
For example, if the API URL is “`http://demoapi.egame.spadegaming.com/api`”, then the picture full path is “`http://demoapi.egame.spadegaming.com/thumbnail/aa.jpg`”
- The API is used for background job. For our API is server to server, the pictures cannot access for public IP. We recommend run the job every day or every week, to download the pictures and display in website.

getGames(Request and response messages example):

JSON getGamesRequest:

```
{
  "merchantCode": "SPADE",
  "serialNo": "20130502191551906534"
}
```

XML getGamesRequest:

```
<getGamesRequest>
  <serialNo>20130502191551906534</serialNo>
  <merchantCode>SPADE</merchantCode>
</getGamesRequest>
```

JSON getGamesResponse:

```
{
  "games": [
    {
      "gameCode": "S-DG02",
      "gameName": "DerbyNight",
      "jackpot": true,
      "thumbnail": "/thumbnail/S-DG02.jpg",
      "screenshot": "/screenshoot/S-DG02.jpg",
      "jackpotCode ": "Holy",
      "jackpotName ": "Holy Jackpot"
    }
  ],
  "merchantCode": "SPADE",
  "code": 0,
  "msg": "success",
  "serialNo": "20130502191551906534"
}
```

XML getGamesResponse:

```
<getGamesResponse>
  <games>
    <gameCode>S-DG02</gameCode>
    <gameName>DerbyNight</gameName>
    <jackpot>true</jackpot>
    <thumbnail>/thumbnail/S-DG02.jpg</thumbnail>
    <screenshot>/screenshot/S-DG02.jpg</screenshot>
    <mthumbnail_>http://xxxx.com/aa.jpg</mthumbnail_>
    <jackpotCode_>Holy</jackpotCode_>
    <jackpotName_>Holy Jackpot</jackpotName_>
  </games>
  <merchantCode>SPADE</merchantCode>
  <code>0</code>
  <msg>success</msg>
  <serialNo>20130502191551906534</serialNo>
</getGamesResponse>
```

4.11 Force Logout for online members

Usage: Kick selected user or all users out of E-games

Interface Name: **kickAcct**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **KickAcctRequest**:

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	No	SPADEPLAYER1	Player ID, Leave empty to force all “online” users to log out of E-games/Lottery.

Response message definition for **KickAcctResponse**:

Name	Data Type	Mandatory	Example	Description
List	String[]	Yes	[SPADEPLAYER1]	List of members that have been kicked out.

KickAcct(Request and response messages example):

JSON KickAcctRequest:

```
{
  "acctId": "TESTPLAYER1",
  "merchantCode": "SPADE",
  "serialNo": "20130430164644935780"
}
```

XML KickAcctRequest:

```
<KickAcctRequest>
  <serialNo>20130430164644935780</serialNo>
  <merchantCode>SPADE</merchantCode>
  <acctId>TESTPLAYER1</acctId>
</KickAcctRequest>
```

JSON KickAcctResponse:

```
{
  "list": [
    "TESTPLAYER1"
  ],
  "merchantCode": "SPADE",
  "code": 0,
  "msg": "success",
  "serialNo": "20130430164644935780"
}
```

XML KickAcctResponse:

```
<KickAcctResponse>
  <serialNo>20130430164644935780</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <list>
    <string>TESTPLAYER1</string>
  </list>
</KickAcctResponse>
```

4.12 Create Token

Usage: Merchant call createToken API to get token and proceed redirect to ticket details page.

Interface Name: **createToken**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **createTokenRequest**:

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code
action	Varchar(50)	Yes	ticketLog	Merchant's action
serialNo	Varchar(50)	Yes	20120723175139440637	Serial No

Response message definition for **createTokenResponse**:

Name	Data Type	Mandatory	Example	Description
token	Varchar(80)	Yes	fe1a85adc54545d29	
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code
serialNo	Varchar(50)	Yes	20120723175139440637	Serial No

Note : This function is not for use in created the token of authorize used.

createToken (Request and response messages example)

JSON createTokenRequest:

```
{
  "acctId": "TESTPLAYER1",
  "merchantCode ": "SPADE",
  "action": "ticketLog",
  "serialNo": "20120722224255982841"
}
```

XML createTokenRequest:

```
<createTokenRequest>
  <acctId>TESTPLAYER1</acctId>
  <merchantCode>SPADE</merchantCode>
  <action>ticketLog</action>
  <serialNo>20120723161423158480</serialNo>
</createTokenRequest>
```

JSON createTokenResponse:

```
{
  "token": "fela3ret56dhhT999",
  "merchantCode": "SPADE",
  "serialNo": "20120722224255982841"
}
```

XML createTokenResponse:

```
<createTokenResponse>
  <token>fela3ret56dhhT999</token>
  <merchantCode>SPADE</merchantCode>
  <serialNo>20120722224255982841</serialNo>
</createTokenResponse>
```

4.13 Ticket Details Page

Usage: Player view ticket details by clicking *View ticket details* button found in merchant website.

Interface Name: **Redirect To Ticket Details Page**

API Provider: Game Provider

API Requestor: Merchant

Name	Data Type	Mandatory	Example	Description
ticketId	Varchar(20)	Yes	641482277	Ticket number.
acctId	Varchar(50)	Yes	TESTPLAYER1	Player's unique Id
action	Varchar(50)	Yes	ticketLog	Merchant's action
token	Varchar(80)	Yes	fe1a85adc54545d29	
language	Varchar(10)	No	en_US	Language to display ticket details page.

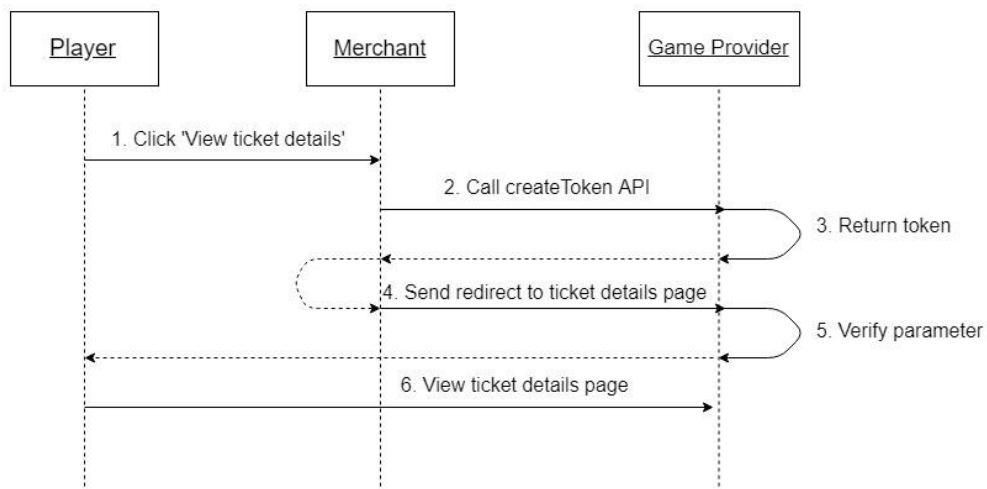
Note:

1. For language, Refer to appendix 2.
2. For action, using 'ticketLog' for ticket details page

Ticket Details Redirect URL and Parameters:

http://portal.e-games.com/merchantCode/createToken/?ticketId=248114533&acctId=TESTPLAYER1&action=ticketLog&token=fe1a85adc54545d2963b661a22d09c9e&language=en_US

Get Ticket Details Flow



4.14 Query Exist of player Acct

Usage: Merchant wishes to check player Acct whether exist or not.

Interface Name: **checkAcct**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **CheckAcctRequest**:

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
includeType	Boolean	No	True	Confirming is “youtuber” player or not
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code
serialNo	Varchar(50)	Yes	20120722231413595602	Generated by merchant to indicate message sequence.

Response message definition for **CheckAcctResponse**:

Name	Data Type	Mandatory	Example	Description
serialNo	Varchar(50)	Yes	20120722224255982841	Generated by merchant to indicate message sequence.
code	Integer	Yes	0	Processed result status.
acctType	Integer	No	0	The definition of code 0 : normal 1 : youtuber
msg	Varchar(128)	Yes	success	Processed result description.
merchantCode	Varchar(10)	Yes	SPADE	Unique code that identifies the Merchant.

checkAcct Request and response messages example:

JSON checkAcct Request:

```
{
  "acctId": "TESTPLAYER1",
  "includeType": true,
  "merchantCode": "SPADE",
  "serialNo": "20180722231413699735"
}
```

XML checkAcct Request:

```
<CheckAcctRequest>
  <acctId>TESTPLAYER1</acctId>
  <includeType>true</includeType>
  <merchantCode>SPADE</merchantCode>
  <serialNo>20180722231413699735</serialNo>
</CheckAcctRequest>
```

JSON checkAcct Response:

```
{
  "serialNo": "20180722231413699735",
  "code": 0,
  "acctType": 0,
  "msg": "Success",
  "merchantCode": "SPADE"
}
```

XML checkAcct Response:

```
<CheckAcctResponse>
  <serialNo>20180722231413699735</serialNo>
  <code>0</code>
  <acctType>0</acctType>
  <msg>Success</msg>
  <merchantCode>SPADE</merchantCode>
</CheckAcctResponse>
```

4.15 Query player daily summary by game

Usage: Search for player daily summary by each game

Interface Name: **playerDailySumByGame**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **playerDailySumByGameRequest**:

Name	Data Type	Mandatory	Example	Description
beginDate	Char(15)	Yes	20120721T000000	Start date range to search. Format must be in yyyyymmdd'T'24hhmmss
endDate	Char(15)	Yes	20120721T010000	End date range to search. Format must be in yyyyymmdd'T'24hhmmss
pageIndex	Integer	Yes	1	Page number
serialNo	Varchar(50)	Yes	20120722231413595602	Generated by merchant to indicate message sequence.
merchantCode	Varchar(50)	Yes	SPADE	Merchant Code
acctId	Varchar(50)	No	TESTPLAYER1	Player's unique Id

Response message definition for **playerDailySumByGameResponse**:

Name	Data Type	Mandatory	Example	Description
resultCount	Integer	Yes	100	Total record count
pageCount	Integer	Yes	1	Total page count
list	PlayerBetInfo[]	Yes		PlayerBetInfo Array

PlayerBetInfo definition:

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	Yes	TESTPLAYER1	Player ID
list	BetInfo[]	Yes		BetInfo Array

BetInfo definition:

Name	Data Type	Mandatory	Example	Description
bets	Integer	Yes	1	Bet Count
betAmount	Decimal(20,4)	Yes	20	Bet Amount
winLoss	Decimal(18,6)	Yes	-20	Member win loss.
jackpotAmount	Decimal(18,6)	Yes	0	
jpWin	Decimal(18,6)	Yes	-100.000000	Jackpot Win
categoryId	Varchar(10)	Yes	SM or TB or AD or BN	Game Category
gameCode	Varchar(10)	Yes	S-GD02	Game Code

Notes :

1. Maximum retrieve interval is 1 day.
2. Retrieve data datetime must be 12 hours ago from current datetime.
3. Data query by hour

playerDailySumByGame (Request and response messages example):

JSON playerDailySumByGameRequest:

```
{
  "beginDate": "20120721T000000",
  "endDate": "20120721T010000",
  "pageIndex": 1,
  "merchantCode": "SPADE",
  "serialNo": "20120723175139440637"
}
```

XML playerDailySumByGameRequest:

```
<PlayerDailySumByGameRequest>
  <serialNo>20120723175139440637</serialNo>
  <merchantCode>SPADE</merchantCode>
  <beginDate>20120721T000000</beginDate>
  <endDate>20120721T010000</endDate>
  <pageIndex>1</pageIndex>
</PlayerDailySumByGameRequest>
```

JSON playerDailySumByGameResponse:

```
{
  "resultCount": 2,
  "pageCount": 1,
  "list": [
    {
      "acctId": "Test10001",
      "list": [
        {
          "bets": 34,
          "betAmount": 20,
          "winLoss": 30,
          "jackpotAmount": 1.5,
          "jpWin": -100.0,
          "categoryId": "SM",
          "gameCode": "S-GD02"
        }
      ]
    },
    {
      "acctId": "Test10002",
      "list": [
        {
          "bets": 34,
          "betAmount": 20,
          "winLoss": 30,
          "jackpotAmount": 1.5,
          "jpWin": -100.0,
          "categoryId": "SM",
          "gameCode": "S-GD02"
        }
      ]
    }
  ],
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120723175139440637"
}
```

XML playerDailySumByGameResponse:

```
<PlayerDailySumByGameResponse>
  <serialNo>20120723175139440637</serialNo>
  <code>0</code>
  <msg>success</msg>
  <merchantCode>SPADE</merchantCode>
  <resultCount>1</resultCount>
  <list>
    <PlayerBetInfo>
      <acctId>Test10001</acctId>
      <list>
        <BetInfo>
          <bets>20</bets>
          <betAmount>20</betAmount>
          <winLoss>30</winLoss>
          <jackpotAmount>0</jackpotAmount>
          <jpWin>-100.000000</jpWin>
          <categoryId>SM</categoryId>
          <gameCode>S-GD02</gameCode>
        </BetInfo>
      </list>
    </PlayerBetInfo>
  </list>
</PlayerDailySumByGameResponse>
```

4.16 Set Youtuber

Usage: When merchant wants to set youtuber feature for a particular member.

Interface Name: **setYoutuber**

API Provider: Game Provider

API Requestor: Merchant

Request Message Definition for **setYoutuberRequest**:

Name	Data Type	Mandatory	Example	Description
acctId	Varchar(50)	Yes	TESTPLAYER1	Unique player ID
currency	Char(3)	Yes	USD	Currency Code
youtuber	Boolean	Yes	True	True: Youtuber False: Normal
merchantCode	Varchar(50)	Yes	SGDEMO	Merchant Code
serialNo	Varchar(50)	Yes	20120722231413595602	generated by "requestor"

Response message definition for **setYoutuberResponse**:

Name	Data Type	Example	Description
serialNo	Varchar(50)	20120722224255982841	Generated by merchant to indicate message sequence.
code	Integer	0	Processed result status.
msg	Varchar(128)	success	Processed result description.
merchantCode	Varchar(10)	SGDEMO	Unique code that identifies the Merchant.

setYoutuber (Request and response messages example):

JSON setYoutuberRequest:

```
{
  "acctId": "TESTPLAYER1",
  "currency": "USD",
  "youtuber": true,
  "merchantCode": "SPADE",
  "serialNo": "20120722231413699735"
}
```

XML setYoutuberRequest:

```
<setYoutuberRequest>
  <acctId>TESTPLAYER1</acctId>
  <currency>USD</currency>
  <merchantCode>SPADE</merchantCode>
  <serialNo>20120722231413699735</serialNo>
  <youtuber>true</youtuber>
</setYoutuberRequest>
```

JSON setYoutuberResponse:

```
{
  "merchantCode": "SPADE",
  "msg": "success",
  "code": 0,
  "serialNo": "20120722231413699735"
}
```

XML setYoutuberResponse:

```
<setYoutuber>
  <code>0</code>
  <merchantCode>SPADE</merchantCode>
  <msg>success</msg>
  <serialNo>20120722231413699735</serialNo>
</setYoutuber>
```

Appendix 1 Response Code Definition

Usage: The response code that Merchant will receive are defined through the API.

Code	Message in English	Description
0	Success	Interface call is successful.
1	System Error	Internal system error at Game provider. Example: Program bug or database connection failed.
2	Invalid Request	Request is not support by Game provider.
3	Service Inaccessible	Game provider API down.
100	Request Timeout	
101	Call Limited	
104	Request Forbidden	
105	Missing Parameters	
106	Invalid Parameters	
107	Duplicated Serial NO.	
109	Related id not found	This Transfer API is usually used for a single wallet
110	Record ID Not Found	
111	Duplicated request	
112	API Call Limited	Exceed the limit of calling API
113	Invalid Acct ID	Acct ID incorrect format
118	Invalid Format	Parse Json Data Failed
120	IP no whitelisted	
5003	System Maintenance	
10113	Merchant Not Found	
10116	merchant suspend	
50099	Acct Exist	
50100	Acct Not Found	
50101	Acct Inactive	
50102	Acct Locked	
50103	Acct Suspend	
50104	Token Validation Failed	
50110	Insufficient Balance	
50111	Exceed Max Amount	
50112	Currency Invalid	Deposit/withdraws currency code are not consistent with Player's default currency code or merchant does not use the currency code.
50113	Amount Invalid	Deposit/withdraw Amount must be greater than 0.
50115	Date Format Invalid	Date Format incorrect format

Appendix 2 Language Code Definition

ISO Language Code	Description
en_US	English
zh_CN	Simplified Chinese
th_TH	Thai
id_ID	Indonesian
vi_VN	Vietnamese
ko_KR	Korea
ja_JP	Japan
ru_RU	Russia
tr_TR	Turkey
es_ES	Spanish
fr_FR	French

Appendix 3 Currency Code Definition (ISO)

ISO Currency Code	Description	ISO Currency Code	Description
BDT	Bangladeshi Taka	MXN	Mexican Peso
CNY	Chinese Yuan (Renminbi)	NZD	New Zealand Dollar
USD	US Dollar	PEN	Peruvian New Sol
EUR	Euro	PLN	Polish Zloty
EPV	Euro	PYG	Paraguayan Guarani
GBP	British Pound	RSD	Serbian Dinar
HKD	Hong Kong Dollar	UYU	Uruguayan Peso
SGD	Singapore Dollar	ZAR	South African Rand
KRW	South Korean Won	UAH	Ukrainian hryvnia
JPY	Japanese Yen	TND	Tunisian Dinar
THB	Thai Baht	RUB	Russian ruble
IDR	Indonesia Rupiah	TRY	Turkish Lira
ID2	Indonesia Rupiah	SEK	Swedish Krona
INR	Indian Rupee	NOK	Norwegian Krone
VND	Viet Nam Dong	AED	Dirham
VN2	Viet Nam Dong	AMD	Armenian Dram
MYR	Malaysia Ringgit	CAD	Canadian Dollar
MMK	Myanmar Kyat	CHF	Swiss franc
MMV	Myanmar Kyat	CLP	Chilean Peso
AUD	Australian Dollar	CZK	Czech Koruna
ALL	Albania	DKK	Danish Krone
BRL	Brazilian Real	ARS	Argentine Peso
BGN	Bulgarian Lev	COP	Colombian Peso
IRR	Iranian Rial	CO2	Colombian Peso
IR2	Iranian Rial	PKR	Pakistan Rupee
KZT	Kazakhstani Tenge	HUF	Hungarian Forint
XAF	Central African CFA franc	XOF	West African CFA franc
AZN	Azerbaijani Manat	NGN	Nigerian Naira
ILS	Israeli Shekel	LBP	Lebanese pound
MDL	Moldovan leu	ETB	Ethiopian birr
TJS	Tajikistani somoni	BAM	Bosnia-Herzegovina Convertible Marka
KGS	Kyrgyzstani som	TMT	Turkmenistani manat
BND	Bruneian Dollars	DEM	Deutsche Mark
KSH	Kenyan shilling	NPR	Nepalese rupee
BYN	Belarusian ruble	RON	Romanian leu

GEL	Georgian lari	KHR	Cambodian Riels
LAK	Lao Kip	BIF	Burundian Francs

Notes:

1. IRR, COP, VND, MMV, LAK, KHR and IDR currency using ratio 1:1000
2. IR2, CO2, VN2, MMK and ID2 currency using ratio 1:1

Appendix 4 API Sample Codes in Java(HTTP)

Constants.java

```
public interface Constants {
    public static final Charset Charset = Charsets.UTF_8;
    public static final String DateFormat = "yyyyMMdd'T'HHmmss";
    public static final String GZIP = "gzip";
    public static final int PageSize = 100;

    public static interface HttpHeaders {
        public static final String ACCEPT_ENCODING =
            "Accept-Encoding";
        public static final String ACCEPT_LANGUAGE =
            "Accept-Language";
        public static final String CONTENT_LENGTH =
            "Content-Length";
        public static final String CONTENT_TYPE = "Content-Type";
        public static final String CONTENT_ENCODING =
            "Content-Encoding";
        public static final String API = "API";
        public static final String DATA_TYPE = "DataType";
        public static final String DIGEST = "Digest";
    }

    public static interface DataType {
        public static final String xml = "XML";
        public static final String Json = "JSON";
    }
}
```

Api Post.java

```
public static byte[] post(String target, String api, String
dataType, byte[] data) throws Exception {
    String requestHash = DigestUtils.digest( data );
    HttpURLConnection conn = null;
    OutputStream out = null;
    InputStream in = null;
    try {
        String dataLength = Integer.toString( data.length );
```

```
//Create connection
URL url = new URL( target );
conn = (URLConnection) url.openConnection();
conn.setRequestMethod( "POST" );
conn.setRequestProperty( API, api );
conn.setRequestProperty( DATA_TYPE, dataType );
conn.setRequestProperty( DIGEST, requestHash );
conn.setRequestProperty( CONTENT_LENGTH, dataLength );
conn.setRequestProperty( ACCEPT_ENCODING, GZIP );
conn.setUseCaches( false );
conn.setDoInput( true );
conn.setDoOutput( true );
conn.setReadTimeout( timeout );

//Send request & handle Response
out = conn.getOutputStream();
out.write( data );
out.flush();
in = conn.getInputStream();
return handleResponse( in, conn );
} finally {
    IOUtils.closeQuietly( out );
    IOUtils.closeQuietly( in );
    if ( conn != null ) {
        conn.disconnect();
    }
}
}

private static byte[] handleResponse(final InputStream in,
final HttpURLConnection conn) throws IOException {
    final int code = conn.getResponseCode();
    if ( code != 200 ) {
        throw new RuntimeException( "server return error! status "
+ code );
    }
    InputStream is = in;
    String encoding = conn.getHeaderField( CONTENT_ENCODING );
    if ( StringUtils.isEmpty( encoding ) &&
encoding.indexOf( GZIP ) >= 0 ) {
        int contentLength = conn.getContentLength();
        is = contentLength > 0 ? new GZIPInputStream( in,
contentLength ) : new GZIPInputStream( in );
    }
}
```

```
}  
byte[] data = IOUtils.toByteArray( is );  
String responseHash = conn.getHeaderField( DIGEST );  
if ( StringUtils.isEmpty( responseHash ) ) {  
    String computedHash = DigestUtils.digest( data );  
    if ( !responseHash.equals( computedHash ) ) {  
        String msg = String.format( "received digest data  
invalid! response: %s", new String( data, Charset ) );  
        throw new RuntimeException( msg );  
    }  
}  
return data;  
}
```


Appendix 5 API Sample Codes in C#(HTTP)

HttpHeader.cs

```
public class HttpHeaders {  
    public const String AcceptEncoding = "Accept-Encoding";  
    public const String AcceptLanguage = "Accept-Language";  
    public const String ContentEncoding = "Content-Encoding";  
    public const String Api = "API";  
    public const String DataType = "DataType";  
    public const String Digest = "Digest";  
}
```

HttpPost.cs

```
public class HttpPost  
{  
    public event MessageEventHandler OnMessageReceived;  
  
    private static readonly HashAlgorithm DigestProvider =  
        new MD5CryptoServiceProvider();  
    private static readonly Encoding Charset = Encoding.UTF8;  
  
    public void Send(String target, String api, String  
        dataType, String data)  
    {  
        Byte[] body = Charset.GetBytes(data);  
        String hash = ComputeHash(body);  
  
        HttpWebRequest request =  
            (HttpWebRequest)WebRequest.Create(target);  
        request.Method = "POST";  
        request.ContentLength = body.Length;  
        request.Headers[HttpHeaders.Api] = api;  
        request.Headers[HttpHeaders.DataType] = dataType;  
        request.Headers[HttpHeaders.AcceptEncoding] =  
            "gzip"; //support gzip  
        request.AutomaticDecompression =  
            DecompressionMethods.GZip;  
        request.Headers[HttpHeaders.Digest] = hash;  
        using (Stream os = request.GetRequestStream())  
        {  
            os.Write(body, 0, body.Length);  
            os.Flush();  
        }  
    }  
}
```

```
        os.Write(body, 0, body.Length);
    }

    IAsyncResult result = request.BeginGetResponse(new
    AsyncCallback(AsyncReceive), request);
}

private void AsyncReceive(IAsyncResult result)
{
    HttpRequest request =
    (HttpRequest)result.AsyncState;
    HttpResponse response =
    (HttpResponse)request.EndGetResponse(result);
    string message = null;
    using (Stream stream = response.GetResponseStream())
    {
        StreamReader sr = new StreamReader(stream,
        Charset);
        message = sr.ReadToEnd().Trim();

        if (response.StatusCode != HttpStatusCode.OK)
        {
            message += "err: " + response.StatusCode +
            Environment.NewLine;
        }
    }

    string hash = response.GetResponseHeader("hash");
    if (!String.IsNullOrEmpty(hash)
    && !"err".Equals(message))
    {
        Byte[] body = Charset.GetBytes(message);
        String computed = ComputeHash(body);
        if (!computed.Equals(hash))
        {
            message += String.Format("hash not equal:
            \nreceived: {0}, computed: {1}", hash,
            computed);
        }
        else
        {
            hash += Environment.NewLine + "Hash check
            passed";
        }
    }
}
```

```
    }  
    }  
  
    if (OnMessageReceived != null)  
    {  
        MessageEventArgs e = new MessageEventArgs();  
        e.Data = message;  
        OnMessageReceived(this, e);  
    }  
}  
  
private String ComputeHash(Byte[] body)  
{  
    Byte[] hashBytes = DigestProvider.ComputeHash(body);  
    String hash =  
        BitConverter.ToString(hashBytes).Replace("-",  
        String.Empty);  
    return hash;  
}  
}
```

Appendix 6 Authorize Example Code (merchant)

```
public class AuthorizeAction implements ResponseCodes {
    //dependent inject from config
    private String merchant;

    //dependent inject
    private TokenManager tokenManager;
    private AcctDao acctDao;

    public AuthorizeResponse execute(AuthorizeRequest request) {
        AuthorizeResponse response = new AuthorizeResponse();
        if ( !merchant.equals( request.getMerchantCode() ) ) {
            response.setCode( MERCHANT_NOT_FOUND );
            return response;
        }
        final String acctId = request.getAcctId();
        final String reqToken = request.getToken();
        String token = tokenManager.findTokenById( acctId );
        if ( isEmpty( token ) || !token.equals( reqToken ) ) {
            response.setCode( TOKEN_INVALID );
            return response;
        }
        //expire token and enhance security
        tokenManager.expireToken( acctId );

        Acct acct = acctDao.findById( acctId );
        if(acct == null){
            response.setCode( ACCT_NOT_FOUND );
            return response;
        }

        AcctInfo info = new AcctInfo();
        info.setAcctId( acct.getId() );
        info.setUserName( acct.getName() );
        info.setCurrency( acct.getCurrency() );
        response.setAcctInfo( info );

        response.setCode( SUCCESS );
        return response;
    }
}
```

Appendix 7 RSA Decryption Example

```
import java.security.Key;
import java.security.KeyFactory;
import java.security.Provider;
import java.security.interfaces.RSAPrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import javax.crypto.Cipher;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public static void decrypt(String privateKey,String
encryptedData ) throws Exception {
    Provider provider = new BouncyCastleProvider( );
    byte[] input =
Hex.decodeHex( encryptedData.toCharArray( ) );
    Cipher cipher = Cipher.getInstance( "RSA", provider );
    byte[] hex = Hex.decodeHex( privateKey.toCharArray( ) );
    PKCS8EncodedKeySpec pkcs8KeySpec = new
PKCS8EncodedKeySpec( hex );
    KeyFactory keyFactory = KeyFactory.getInstance( "RSA",
provider );
    Key privateK = keyFactory.generatePrivate( pkcs8KeySpec );

    cipher.init( 2, ( RSAPrivateKey ) privateK );

    byte[] raw = cipher.doFinal( input );

    String s = new String( raw, "UTF-8" );
    String result = new
StringBuilder( s ).reverse( ).toString( );
    System.out.println( result );
}
```