

Version	Date	Changes
1.00	25 Feb 2026	First and Initial version of APIs
1.01	26 Feb 2026	Add <code>get_player_balance</code> API
1.02	02 Apr 2026	Add <code>get_tables</code> API. Add <code>join_table_seat</code> API. Add <code>update_seat_ping</code> API Add <code>create_player_action</code> API Add <code>leave_table_seat</code> API
1.03	22 Apr 2026	Add <code>id</code> and <code>current_round_no</code> in <code>get_tables</code> API. Add <code>get_round</code> API. Add <code>get_player_hand</code> API.
1.04	12 May 2026	Add Bet into Player Action Type Add details on Web Socket and Events
1.05	26 July 2026	Update <code>join_table_seat</code> API to return success when Player already in the seat. Add <code>get_player_seats</code> API.
1.06	8 Sept 2026	Add <code>get_table_by_id</code> API. Add <code>get_player_hand_history</code> API Update and standardized <code>table_info</code> . Add <code>current_bet</code> into <code>table_info</code> . Add Websocket events when deal card to flop, turn and river.

Poker API

Index

Table 1: All Available APIs

[illegible]

Overview

Domain	https://api.wonpoker.online/
Base URL	https://api.wonpoker.online/api/
pkey	<i>To be provided</i>
skey	<i>To be provided</i>

The complete API URL is **[Base URL] + [URL Name]**. Example to perform login, the complete API URL is https://api.wonpoker.online/api/player_login.

Every API need to be requested using POST method.

Every API will return the result in JSON format.

Every request needs to append with these parameters:

Table 2: Mandatory Parameters for all API Requests

Parameters	Type	Max Length	Required	Remark
pkey	string	16	Yes	The Public Key provided
time	float	-	Yes	Current Epoch time when making the request. Each Epoch time must be unique and cannot offset more than 120 seconds. Up to 2 decimal places is allowed.
signature	string	32	Yes	Can be generated using formula: md5 (time + skey)
player id	integer	-	Yes*	* Required only on Authenticated URLs for player (Refer to Table 1)
player token	string	32	Yes*	* Required only on Authenticated URLs for player (Refer to Table 1)

Assume your skey is **P2B5jwQXiw60I5ic9s8MeWniG1o9GteS** and current Unix Epoch time, time is **1564895669.02**, hence your signature can be obtained using:

```
md5 (1564895669.02P2B5jwQXiw60I5ic9s8MeWniG1o9GteS)
```

Result is:

```
dacf1a335219b1a19e8ffc7c4b48d436
```

(Convert to lowercase)

Url: create_player_action

This is to let player make his/her necessary action during the game.

Request

Parameters	Type	Max Length	Required	Remark
table_id	integer	-	Yes	
seat_index	integer	-	Yes	
action	char	1	Yes	Refer to Appendix A: Player Action Type
amount	float	-	Yes / No	Required only when action RAISE

Response

Parameters	Remark
player_id	The unique ID of the player
balance	The current balance of the player
table_balance	The current table balance of the player
round_no	The current Round No
pot_total	Total Amount of the current pot
action	The Action sent
status	If everything ok, this will show “Success”

Sample of the responses:

```
{
  "response": {
    "player_id": 2,
```

```
"balance": "100400.0000",  
"table_balance": "200.0000",  
"round_no": "T00320260402-235",  
"pot_total": "5.0000",  
"action": "F",  
"status": "SUCCESS"  
}  
}
```

Url: get_player_balance

This is to get the player latest balance and other information.

Request

Parameters	Type	Max Length	Required	Remark

Response

Parameters	Remark
player_id	The unique ID of the player
player_type	The type of player. Refer to Appendix A: Player Type for more details.
username	The registered username of the player
balance	The current balance of the player

Sample of the responses:

```
{
  "response": {
    "player_id": 2,
    "player_type": "G",
    "username": "easy001",
    "balance": "100000.0000"
  }
}
```

Url: get_player_hand

This is to get the details of the player hand.

Request

Parameters	Type	Max Length	Required	Remark
round_no	String	64	Yes	

Response

Parameters	Remark
table_code	The table code of the round.
round_no	The no of the round
seat_index	The seat index of the player
player_id	Player's ID
player_username	Player's username
card_1	Player's Card 1
card_2	Player's Card 2

Sample of the responses:

```
{
  "response": {
    "table_code": "T003",
    "round_no": "T00320260422-132",
    "seat_index": 1,
    "player_id": 2,
```

```
"player_username": "easy001",  
"card_1": "S07",  
"card_2": "S04"  }  
}
```

Url: get_player_hand_history

This is to get the details of the player hand.

Request

Parameters	Type	Max Length	Required	Remark
table_id	Integer	32	No	If no table_id provided, hand history from all tables will be shown.
limit	Integer	-	No	If no limit provided, default will show up to 100 historical records.

Response

Parameters	Remark
table_code	The table code of the round.
round_no	The no of the round
seat_index	The seat index of the player
player_id	Player's ID
player_username	Player's username
card_1	Player's Card 1
card_2	Player's Card 2

Sample of the responses:

```
{
  "response": [
    {
      "table_code": "T002",
      "round_no": "T00220260908-475",
```

```
    "seat_index": 2,  
    "player_id": 4,  
    "player_username": "easy003",  
    "card_1": "C11",  
    "card_2": "C13"  
  }  
  ]  
}
```

Url: get_player_seats

This is to get the details of all the seats that the current player had joined.

Request

Parameters	Type	Max Length	Required	Remark

Response

Parameters	Remark
table_id	The ID of the table
table_code	The Code of the table.
table_play_type	The play type of the table. Refer to Appendix A: Table Play Type for more details.
table_seat_count	The total seats of the table
table_sb	The small blind of the table
table_bb	The big blind of the table
table_round_no	The table's current round no
seat_index	The seat index of the player

Sample of the responses:

```
{
  "response": {
    "seats": [
      {
        "table_id": 2,
        "table_code": "T002",
```

```
"table_play_type": "F",
"table_seat_count": 8,
"table_sb": "1.0000",
"table_bb": "2.0000",
"table_round_no": "T00220260726-95",
"seat_index": 1
},
{
  "table_id": 3,
  "table_code": "T003",
  "table_play_type": "F",
  "table_seat_count": 8,
  "table_sb": "2.0000",
  "table_bb": "3.0000",
  "table_round_no": "T00320260617-111",
  "seat_index": 1
}
]
}
```

Url: get_round

This is to get the details of the round.

Request

Parameters	Type	Max Length	Required	Remark
round_no	String	64	Yes	

Response

Parameters	Remark
table_code	The table code of the round.
no	The no of the round
start_at	Round starting time
end_at	Round ending time
flop_1	Card at Flop 1
flop_2	Card at Flop 2
flop_3	Card at Flop 3
flop_updated_at	Date time when Flop Card deal
turn	Card at Turn
turn_updated_at	Date time when Turn Card deal
river	Card at River
river_updated_at	Date time when River Card deal
total_pot	Total Pot Amount
pot_updated_at	Date time of last pot amount updated
button_seat_index	The Seats index of button at current round
sb_seat_index	The Seats index of small blind at current round
bb_seat_index	The Seats index of big blind at current round

seat_updated_at	Date time when the Seats is determined.
status	Status of the Round. Refer to Appendix A: Round Status for more details.

Sample of the responses:

```
{
  "response": {
    "table_code": "T003",
    "no": "T00320260422-41",
    "start_at": "2026-04-22T13:39:41.000000Z",
    "end_at": "2026-04-22T13:40:42.000000Z",
    "flop_1": null,
    "flop_2": null,
    "flop_3": null,
    "flop_updated_at": null,
    "turn": null,
    "turn_updated_at": null,
    "river": null,
    "river_updated_at": null,
    "total_pot": "5.0000",
    "pot_updated_at": "2026-04-22T13:40:21.000000Z",
    "button_seat_index": 1,
    "sb_seat_index": 1,
    "bb_seat_index": 2,
    "seat_updated_at": null,
    "status": "C"
  }
}
```

Url: get_table_by_id

This is to get table based on Table ID.

Request

Parameters	Type	Max Length	Required	Remark
table_id	Integer	-	Yes	

Response

Refer to [Appendix C: Standard Table Info](#) for more details.

Url: get_tables

This is to get all the tables on the system based on the player type.

Request

Parameters	Type	Max Length	Required	Remark

Response

Refer to [Appendix C: Standard Table Info](#) for more details.

Url: join_table_seat

This is to get all the tables on the system based on the player type. If the player already inside the table, it cannot join again except for current seat.

Request

Parameters	Type	Max Length	Required	Remark
table_id	integer	-	Yes	
seat_index	integer	-	Yes / No	If table seat assign type is random, seat_index is not required.
buy_in_amount	float	-	Yes	Amount to buy in for joining the table.

Response

Parameters	Remark
player_id	The unique ID of the player
balance	The current balance of the player
table_id	The unique ID of the table
seat_index	The unique Seat Index of the Table
seat_assign_type	Seat Assignment type. Refer to Appendix A: Table Seat Assign Type for more details
buy_in_amount	The Player Buy-In Amount, which will become the Table Balance of the player

Sample of the responses:

```
{
  "response": {
    "player_id": 2,
    "balance": "100300.0000",
    "table_id": 3,
```

```
    "seat_index": 1,  
    "buy_in_amount": "300"  
  }  
}
```

Url: leave_table_seat

This is let the player to leave the current seat.

Request

Parameters	Type	Max Length	Required	Remark
table_id	integer	-	Yes	
seat_index	integer	-	Yes / No	If table seat assign type is random, seat_index is not required.

Response

Parameters	Remark
player_id	The unique ID of the player
balance	The current balance of the player
status	If everything OK, it will print "SUCCESS"

Sample of the responses:

```
{
  "response": {
    "player_id": 2,
    "balance": "100210.0000",
    "status": "SUCCESS"
  }
}
```

Url: player_login

This is to login a player into the system by getting the authenticated token.

Request

Parameters	Type	Max Length	Required	Remark
username	string	128	Yes	
password	string	128	Yes	

Response

Parameters	Remark
player_id	The unique ID of the player
player_type	The type of player. Refer to Appendix A: Player Type for more details.
token	The token for other authenticated functions
expired_at	The expiry date and time for the token
balance	The current balance of the player

Sample of the responses:

```
{
  "response": {
    "player_id": 1,
    "player_type": "P",
    "token": "lWTHilss25OmPO1rDH1aiD0s2WN39Ie0",
    "expired_at": "2026-02-25 23:33:57",
    "balance": "5000.0000"
```

}
}

Url: player_register

This is to register a player into the system. The registration is for player type “Guest” only.

Request

Parameters	Type	Max Length	Required	Remark
username	string	128	Yes	Must be unique within the system.
password	string	128	Yes	There are some requirements for the password, e.g. minimum length of 6 etc.
name	string	128	Yes	

Response

Parameters	Remark
player_id	The unique ID of the player
player_type	The type of player. Refer to Appendix A: Player Type for more details.
username	The registered username of the player
balance	The current balance of the player

Sample of the responses:

```
{
  "response": {
    "player_id": 2,
    "player_type": "G",
    "username": "easy001",
    "balance": "100000.0000"
  }
}
```

}

Url: update_seat_ping

This is to update the player ping on the seat to allow system to recognize the connectivity of the player in game. It is recommended to update the ping every 10-15 seconds.

Request

Parameters	Type	Max Length	Required	Remark
table_id	integer	-	Yes	
seat_index	integer	-	Yes / No	If table seat assign type is random, seat_index is not required.

Response

Parameters	Remark
player_id	The unique ID of the player
balance	The current balance of the player
table_id	The unique ID of the table
seat_index	The unique Seat Index of the Table
last_ping_at	The latest Ping by the player

Sample of the responses:

```
{
  "response": {
    "player_id": 4,
    "balance": "99700.0000",
    "table_id": 3,
    "seat_index": 2,
```

```
    "last_ping_at": "2026-03-15T05:14:30.000000Z"  
  }  
}
```

Player Side Socket

Overview

Key	8G33mQ.QRs1Tg:vuDyeMp2dvYPzo37bjOudnc2a0-ldiLjxQkGR6IHn-U
Client ID	WONPOKER

The socket using the Ably, the quick start tutorial can refer to: <https://ably.com/docs/getting-started/javascript>

This is the sample JavaScript setup.

```
<script src="https://cdn.ably.com/lib/ably.min-2.js"></script>
<script type="text/javascript">
  async function initAbly() {

    // Initialize the Ably
    const realtimeClient = new Ably.Realtime({
      key: '{{ "ABLY_KEY_SUBSCRIBE" }}',
      clientId: '{{ "ABLY_CLIENT_ID" }}'
    });

    await realtimeClient.connection.once('connected');
    console.log('Ably Connected!');

    // Subscribe to a channel
    const tableUpdateChannel = realtimeClient.channels.get('table-{{ "TABLE_ID" }}-channel');
    await tableUpdateChannel.subscribe((message) => {
      console.log(message.data);
    });

  }
  initAbly();
</script>
```

WebSocket Events

Channel: table-[TABLE_ID]-channel
ROUND_CREATED
This event is being broadcast when a new Round is created.
<pre>{ "event": "ROUND_CREATED", "game_to_start_at": 1778537286, "time": 1778537256.103218, "table_info": { Refer to Appendix C: Standard Table Info for more details. } }</pre>

Channel: table-[TABLE_ID]-channel
SB_BB_DETERMINED
This event is being broadcast when a Position of Small Blind and Big Blind is determined.
<pre>{ "event": "SB_BB_DETERMINED", "time": 1778535529.099221, "table_info": { Refer to Appendix C: Standard Table Info for more details. } }</pre>

Channel: table-[TABLE_ID]-channel
DEAL_CARDS_TO_PLAYERS
This event is being broadcast when a Two Hand Cards is deal to players.
<pre>{ "event": "DEAL_CARDS_TO_PLAYERS", "time": 1778535529.378074, "table_info": { Refer to Appendix C: Standard Table Info for more details. } }</pre>

Channel: table-[TABLE_ID]-channel
PRE_FLOP_INIT_BID
This event is being broadcast when a The Small Blind and Big Blind bet is automatically Bided.
<pre>{ "event": "PRE_FLOP_INIT_BID", "time": 1778535533.04171, "action_seat_index": 1, "bid_amount": "2.0000", "total_pot_amount": 2, "player_id": 2, "player_name": "easy001", "table_info": { Refer to Appendix C: Standard Table Info for more details.} }</pre>

Channel: table-[TABLE_ID]-channel
PRE_FLOP_WAITING_PLAYER_ACTION
This event is being broadcast when a Waiting for the Player to act at Pre-Flop State.
<pre>{ "event": "PRE_FLOP_WAITING_PLAYER_ACTION", "time": 1778535533.441494, "total_pot_amount": 5, "action_seat_index": 1, "player_id": 2, "player_name": "easy001", "action_to_done_by": 1778535553, "table_info": { Refer to Appendix C: Standard Table Info for more details.} }</pre>

Channel: table-[TABLE_ID]-channel
PLAYER_ACTION
This event is being broadcast when the Player make any valid action.
<pre>{ "event": "PRE_FLOP_PLAYER_ACTION", "time": 1778535554.821006, "total_pot_amount": 5, "action_seat_index": 1, "player_id": 2, "player_name": "easy001", "action": "F", "table_info": { Refer to Appendix C: Standard Table Info for more details.} }</pre>

Channel: table-[TABLE_ID]-channel
DEAL_CARDS_TO_FLOP
This event is being broadcast when deal Cards to Flop.
<pre>{ "event": "DEAL_CARDS_TO_FLOP", "time": 1778540649.415979, "card_1": "C01", "card_2": "C04", "card_3": "C05" }</pre>

Channel: table-[TABLE_ID]-channel
DEAL_CARDS_TO_TURN
This event is being broadcast when deal Card to Turn.
<pre>{ "event": "DEAL_CARDS_TO_TURN", "time": 1778540649.415979, "card": "C04" }</pre>

Channel: table-[TABLE_ID]-channel
DEAL_CARDS_TO_RIVER
This event is being broadcast when deal cards to River.
<pre>{ "event": "DEAL_CARDS_TO_RIVER", "time": 1778540649.415979, "card": "C04" }</pre>

Channel: table-[TABLE_ID]-channel
ROUND_COMPLETED
This event is being broadcast when the Round is completed.
<pre>{ "event": "ROUND_COMPLETED", "time": 1778535555.091245, "total_pot_amount": 5, "commission_amount": 0.25, "winner_seat_index": null, "winner_player_id": null, "winner_player_name": null, "table_info": { Refer to Appendix C: Standard Table Info for more details.} }</pre>

Channel: table-player-[PLAYER_TOKEN]-channel
RECEIVE_DEALED_CARDS
This event is being broadcast when the player receives the two hand cards.
<pre>{ "event": "RECEIVE_DEALED_CARDS", "time": 1778540649.415979, "seat_index": 1, "player_id": 2, "player_name": "easy001", "token": "EQirujkqzkDZ47weiGcx2glCqiDJrK21", "card_1": "C01", "card_2": "C04" }</pre>

Appendix A

Card Code

Code	Name	Code	Name
S	Spade	01	A
H	Heart	02 – 10	2 – 10
C	Club	11	J
D	Diamond	12	Q
		13	K

Player Action Type

Code	Name
N	No Action
B	Bet
F	Fold
C	Call
K	Check
R	Raise
A	All-in

Player Type

Code	Name
P	Cash Player
G	Guest

Round Status

Code	Name
O	Open

C	Completed
X	Cancelled
V	Voided

Table Play Type

Code	Name
F	Play for Fun
C	Cash Play
T	Tournament

Table Seat Assign Type

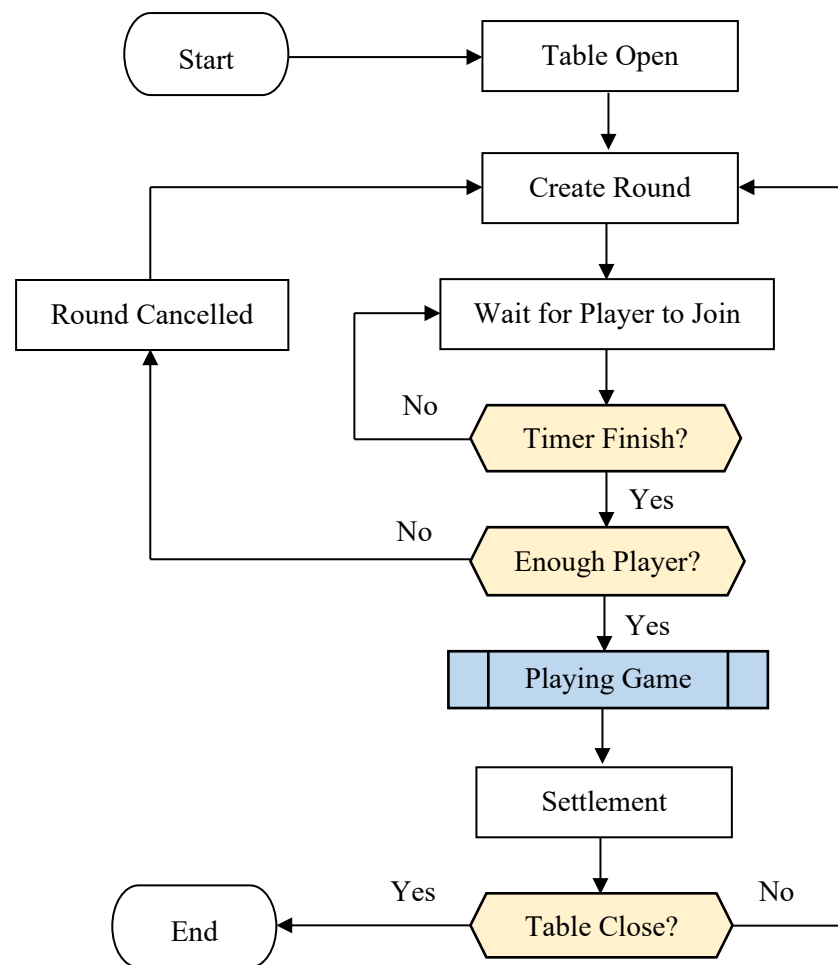
Code	Name
R	Random Assigned
M	Manually Selected

Table Status

Code	Name
O	Opened
P	Playing
S	Settlement
C	Closed

Appendix B

Table Round Flow



Check Game

- 1. Stage 1: Game Started, and before $\text{start_at} + \text{player_join_buffer_time}$**
- 2. Stage 2: after $\text{start_at} + \text{player_join_buffer_time}$, check sb_seat and bb_seat**

Appendix C

Standard Table Info

Parameters		Remark
id		The unique ID of the table.
code		The unique Code of the table.
play_type		The play type of the table. Refer to Appendix A: Table Play Type for more details.
is_open		Is the Table Open? 0 – No, 1 – Yes
seat_count		The maximum seat count for the table
sb		Small Blind
bb		Big Blind
seat_assign_type		Seat Assignment type. Refer to Appendix A: Table Seat Assign Type for more details
next_round_at		Date Time that next Round is going to start. Become Null when Game is playing.
current_round_no		The Current Round No this table is having. Can be null.
player_join_buffer_second		Total Seconds to wait before game start.
player_action_buffer_second		Total Seconds to wait for player's action
min_entry_bb		Minimum amount (calculate by total BB) to bring in when player join.
max_entry_bb		Maximum amount (calculate by total BB) to bring in when player join.
commission		The commission of the table
is_no_flop_no_drop		No Flop no Drop setting
status		Current Status of the Table. Refer to Appendix A: Table Status for more details
seats		The Seats of the table and its attributes
	index	The index of the seat (or seat no)
	is_button	Is this seat currently being Button? 0 – No, 1 – Yes
	is_sb	Is this seat currently being Small Blind? 0 – No, 1 – Yes
	is_bb	Is this seat currently being Big Blind? 0 – No, 1 – Yes
	is_fold	Is this seat currently being fold? 0 – No, 1 – Yes
	is_occupied	Is this seat currently being sit by other player? 0 – No, 1 – Yes
	is_waiting_action	Is this seat currently waiting for player action? 0 – No, 1 – Yes
	last_ping_at	Last Ping received from this seat.
	display_img	The profile image url for the seat
	display_name	The display name for the seat

	display_balance	The current balance for the seat.
	current_bet	The current bet for the seat
Round		The round of the table and its attributes
	table_code	The table code for the round.
	no	The no of the round
	start_at	The date time where current round started
	end_at	The date time where current round ended
	flop_1	The first flop card
	flop_2	The second flop card
	flop_3	The third flop card
	flop_updated_at	The date time where flop cards updated
	turn	The turn card
	turn_updated_at	The date time where turn cards updated
	river	The river card.
	river_updated_at	The date time where river cards updated
	total_pot	The current pot amount
	pot_updated_at	The date time where the pot amount updated
	button_seat_index	The current button seat
	sb_seat_index	The current SB seat
	bb_seat_index	The current BB seat
	seat_updated_at	The date time where the seat is updated
	status	The status of the round. Refer to Appendix A: Round Status for more details
	current_action	Current action for the round.

Sample of the responses:

```
"{
  "id": 2,
  "code": "T002",
  "play_type": "F",
  "is_open": 1,
  "seat_count": 8,
  "sb": "1.0000",
  "bb": "2.0000",
  "seat_assign_type": "R",
  "next_round_at": null,
  "current_round_no": "T00220260908-479",
  "player_join_buffer_second": 30,
  "player_action_buffer_second": 20,
  "min_entry_bb": 40,
  "max_entry_bb": 100,
  "commission": "0.0500",
  "is_no_flop_no_drop": 1,
  "status": "O",
  "seats": [
    {
      "index": 1,
      "is_button": 0,
      "is_sb": 0,
      "is_bb": 0,
      "is_fold": 0,
      "is_occupied": 1,
      "is_waiting_action": 0,
      "last_ping_at": "2026-07-26T02:49:31.000000Z",
      "display_img": null,
      "display_name": "easy001",
      "display_balance": "1000.0000",
      "current_bet": null
    }
  ]
},
```

```
"round": {
  "table_code": "T002",
  "no": "T00220260908-479",
  "start_at": "2026-09-08T08:37:05.000000Z",
  "end_at": null,
  "flop_1": null,
  "flop_2": null,
  "flop_3": null,
  "flop_updated_at": null,
  "turn": null,
  "turn_updated_at": null,
  "river": null,
  "river_updated_at": null,
  "total_pot": null,
  "pot_updated_at": null,
  "button_seat_index": null,
  "sb_seat_index": null,
  "bb_seat_index": null,
  "seat_updated_at": null,
  "status": "O",
  "current_action": "ROUND_CREATED"
}
```