JSON Request Format

General structure

The request format consist of mainly two sections. A *HEADER* provides general information about the Client Software and the table being observed while hand related data is stored in the *GAME ACTIONS* section.

```
{
```

Header

```
"data_source": "dotnet-alpha-scraper",

"source_hand_id": "",
"game_id": "",

"source_table_id": "",
"source_table_name": "",

"seats" : 10,
"hero_seat": 0,

"session_id": "",
"site": "",
"table_id" : 5,
"table_type" : "unknown",
"table_name" : "",
"return_data" : false
```

`data_source` will be of constant value and contain a string that indicates the scraper software used to obtain the table information.

`source_hand_id` and `game_id` will have the same unique value throughout the duration of a hand. A viable candidate is `table_id` concatenated with `DateTime.Now.Ticks` at the beginning of the hand.

`source_table_id` and `source_table_name` will have the same value throughout the lifetime of a table. This value can be generated upon process initialisation.

`seats` Indicates the number of participants in the current hand. The participants will be referred to by their seat-index which must be within `[0 .. seats-1]`

`hero_seat` represents the index of the seat the player is seated on. It must be within the range `[0 .. seats-1]`

`site` A string uniquely identifying the poker site the data is scraped from. A list of valid values will be provided here. If none is available, set it to `dotnet-alpha-scraper-poker`.

`table_id` is a unique integer among all processes. It must be possible to adjust this number through an environment variable that is automatically incremented each time a new process is launched and/ or through a command line switch.

`table_type` is intended to indicated whether it is a heads-up, a six-max, a full-ring or another kind of table. Can safely be set as `unknown`

`table_name` can be set to a random string. The window title would be a fitting candidate.

`return_data` can initially be set to false but must be adjustable through a command line switch or environment variable.

Game action

All hand `actions` are stored in an ordered list/ array of individual action objects.

```
"actions: [
```

- Player data

  The first data block contains information about the players seated at the table.

  ```
  {
      "type": "stack",
      "name": "p_seat_0",
      "uid": "10000",
      "seat": 0,
      "value": 304.25
  },
  {
      "type": "stack",
      "name": "p_seat_1",
      "uid": "10001",
      "seat": 1,
      "value": 2307.93
  },
  {
      "type": "stack",
      "name": "p_seat_2",
      "uid": "10002",
      "seat": 2,
      "value": 584.95
  },
  ```

  `name` and `uid` can be arbitrary values but must be consistent within for the hand. `type` indicates that the json object contains data about player stack sizes and `value` contains the numerical value indicating how much money the player had in his stack before performing committing any chips to the pot.

  `seat` must be strictly sequential. If three players are involved, the indices range from `[0..2]`.

- Mandatory commitments

  Player data is followed by a block of objects indicating actions had player had to perform due to the rules of poker. These include, but are not limited to *Ante*, *Small Blind*, *Big Blind* and *Straddle*.

  – Ante

    There is one ante object added for each player.

    ```
    {
        "type": "ante",
        "seat": 0,
        "value": 1.0
    },
    {
        "type": "ante",
        "seat": 1,
        "value": 1.0
    },
    {
        "type": "ante",
        "seat": 2,
        "value": 1.0
    },
    ```

  - Small Blind

    If a small blind is posted in the hand, an object is added to the list of actions.

    ```
    {
        "type": "sb",
        "seat": 1,
        "value": 2.0
    },
    ```

  - Big Blind

    Then one object is added for the posted big blind.

    ```
    {
        "type": "bb",
        "seat": 2,
        "value": 4.0
    },
    ```

  - Straddle

    If a straddle was posted, an object is added to the list of actions.

    ```
    {
        "type": "str",
        "seat": 0,
        "value": 8.0
    },
    ```

- Preflop (Initial betting round)

  The preflop betting round is implicitly started by adding an object that contains hero's secret cards.

  ```
  {
      "type": "hero_cards",
      "cards": "6d7s"
  },
  ```

  And then followed by a sequence of player actions that occured before the flop.

  ```
  {
      "type": "raise",
      "seat": 0,
      "value": 2.0
  },
  {
      "type": "fold",
      "seat": 1
  },
  {
      "type": "call",
      "seat": 2
  },
  ```

  Valid player actions are *fold*, *check*, *call* and *raise*. The action *bet* is not implemented.

- Postflop (Second, Third and Fourth betting round)

  A postflop start of a postflop betting round is explicitly marked in a separate object.

  - Flop

    ```
    {
        "type": "flop",
        "cards": "4h5dTd"
    },
    ```

  - Turn

    ```
    {
        "type": "turn",
        "cards": "8s"
    },
    ```

  - River

    ```
    {
        "type": "river",
        "cards": "5s"
    },
    ```

  - Postflop Action Sequence

    The object that marks a postflop street as started is followed by a player action sequence similar to preflop. This section will contain a sequence that demonstrates how to generate values for the raise objects.

    ```
    {
        "type": "raise",
        "seat": 0,
        "value": 10.0
    },
    {
        "type": "raise",
        "seat": 2,
        "value": 20.0
    },
    {
        "type": "raise",
        "seat": 0,
        "value": 30.0
    },
    {
        "type": "allin",
        "seat": 2
    },
    {
        "type": "call",
        "seat": 0
    }
    ```

    The collection above maps to the following sequnce:

```
Let the player in Seat 0 be Anna
Let the player in Seat 2 be Bob

Anna raises from  0.0 to 10.0 , hence +10.0
Bob  raises from  0.0 to 20.0 , hence +20.0
Anna raises from 10.0 to 40.0 , hence +30.0
```

`value` indicates the amount the player has taken from his stack and committed to the pot during this action.
The `type` *allin* is preferrable over a *raise* if the player committed all his chips during this action.

The format supports additional `type` such as `show` or `win` which will be omitted in this document.

```
    ]
}
```