

---

# **fut Documentation**

***Release 0.3.10***

**Piotr Staroszczyk**

**Jan 09, 2018**




---

## Contents

---

<b>1</b>	<b>Feature Support</b>	<b>3</b>
<b>2</b>	<b>User Guide</b>	<b>5</b>
2.1	Changelog . . . . .	5
2.2	Installation . . . . .	14
2.3	Introduction . . . . .	14
2.4	Cookbook . . . . .	15
2.5	IDs (object structures?) . . . . .	15
<b>3</b>	<b>API Guide</b>	<b>21</b>
3.1	Squads . . . . .	21
3.2	Squad Building Challenge . . . . .	22
3.3	Transfers . . . . .	22
3.4	Search . . . . .	23
3.5	Transfer List . . . . .	24
3.6	Watch List . . . . .	26
3.7	Store . . . . .	29
3.8	Club . . . . .	29
<b>4</b>	<b>Contributor Guide</b>	<b>33</b>
4.1	Authors . . . . .	33



Release v0.3.10. (*Installation*)  fut is a simple Python library for managing Fifa Ultimate Team. The library enables programmatic access to the official [FUT Web App](#) and [FIFA Companion App](#). If you prefer php language, there is ported version made by InkedCurtis available here: <https://github.com/InkedCurtis/FUT-API>.

The fut library can perform 28 basic [REST](#) functions on your Fifa Ultimate Team. It also includes 6 properties that provide access to various FUT databases (nations, leagues, teams, stadiums, balls, players, playStyles).



# CHAPTER 1

---

## Feature Support

---

- Multi platform (pc, ps3, xbox, ios, android)
- Searching auctions with filters
- Biding
- Selling
- Quick selling
- Full control of watchlist, tradepile and unassigned cards
- Buying and opening packs
- Filling & submitting Squad Battle Challenges (not tested yet)
- Updating credits variable on every request
- Simple keepalive function just to extend session life
- Requesting any card info without login
- Calculating baseID
- Python 2.6-3.6, PyPy





This is basic part of the documentation, it's about installation and importing.

## 2.1 Changelog

### 2.1.1 Changelog

#### 0.3.10 (2017-12-22)

- sku\_a changed into sku\_b (#376 thanks to ricklhp7)

#### 0.3.9 (2017-12-18)

- default page\_size for search 16->36 (thanks to kirov #360)
- add packs method to list all available packs
- fix multiple item quicksell (thanks to farukuzun #366)

#### 0.3.8 (2017-12-07)

- add anti-captcha.com support (all credits go to kirov #353)
- add futbinPrice, futheadPrice
- add missing params to club method (#351)
- tidt & taxv moved to compiled\_2.js

### **0.3.7 (2017-12-01)**

- allow init with None cookies (fix #345 by kirov)
- fix clubConsumables (#347, #348)
- remove path value check in remoteConfig

### **0.3.6 (2017-11-12)**

- add sbsSetChallenges (thanks to dan-gamble #330)
- readme polish (thanks to syndac)
- add tradepileClear
- add sbsSquad
- add sendToSbs
- add clubConsumables
- correct version param in pinevents
- save token between logins (maybe cookies are not needed?)

### **0.3.5 (2017-10-26)**

- various pinEvents improvements
- remove default buy now price for sell method to avoid mistakes
- add buyPack
- add objectives
- add duplicates list
- add level param to club method
- correct tradeStatus params
- check tradeStatus after selling item like webapp do
- add marketDataMaxPrice & marketDataMinPrice to item data parser

### **0.3.4 (2017-10-18)**

- disable debug info

### **0.3.3 (2017-10-18)**

- correct pin values (#314)

### **0.3.2 (2017-10-18)**

- fix syntax error

### 0.3.1 (2017-10-18)

- pinEvents: random timestamp with delay & option request before sending data
- add sbsSets
- correct few details (page\_size, sleep times etc.) - community work :-)
- rename searchAuctions to search (You can still use searchAuctions)
- fix python2 compatibility (#296)
- correct \_ value - all credits go to ricklhp7
- fix sendToWatchlist (jsarasti patch #303)
- proper currentBid check logic (jsarasti patch #303)
- fix squad method (#300)
- fix database (players, nations, leagues, teams, stadiums)
- add balls method (database)
- rewrite searchDefinition (jsarasti work #304)

### 0.3.0 (2017-10-12)

- initial release for fifa 18
- bump useragent
- add ability to login via sms code or totp authenticator (fully automatic)
- pinEvents

### 0.2.19 (2017-09-21)

- searchAuctions: add ability to search rare (special) cards #280
- fix addition request to send code (#285)

### 0.2.18 (2017-05-25)

- do not force log in when not necessary (thanks to xAranaktu #264)
- add missing params, update logic in login (thanks to xAranaktu #266)
- reenable position parsing & add missing keys in item\_data parser (fix #265)
- unify item\_data keys for players & consumables
- add playstyles & stadiums
- add missing param sku\_a (thanks to rafaelget #259)

### **0.2.17 (2017-05-20)**

- fix #262 searchAuctions && piles returns empty list when no results found
- fix wrong fut version in referer on login (thanks to xAranaktu #263)
- init usermassinfo functionality (thanks to xAranaktu #263)
- add tradingEnabled check on login (thanks to xAranaktu #263)

### **0.2.16 (2017-05-17)**

- fix applyConsumable
- add brokeringSku param for tradepile (thanks to pulkitsharma #259)

### **0.2.15 (2017-05-04)**

- huge performance improvement on database load (skip encoding guess)
- fix baseId version calculation
- core: add clubConsumablesDetails
- core: add applyConsumable

### **0.2.14 (2017-04-29)**

- fix player parser

### **0.2.13 (2017-04-28)**

- fix cardInfo for not a player
- cache database

### **0.2.12 (2017-04-28)**

- proper #255 fix - from now we're going to read whole players db on login

### **0.2.10 (2017-04-24)**

- fix baseId calculation (#255)

### **0.2.9 (2017-03-07)**

- proper #250, #251 fix (thanks to bas85)

### 0.2.8 (2017-03-06)

- fix login problems, need confirmation (#250)
- bump fifa version in urls & user-agent etc.
- temporary disabled emulate feature, need more info and work (#250)

### 0.2.7 (2017-01-17)

- fix missing import (#244)

### 0.2.6 (2017-01-10)

- add (minimum request) delay param (#233)
- add fast param to bid method
- use Unauthorized exception (fix #232)

### 0.2.5 (2016-12-28)

- add timeout (#226)

### 0.2.4 (2016-12-15)

- proper(?) page\_size and start values correction (thanks to rafaelget #220)
- fix fut\_home url (thanks to Fanatico1981 #219)

### 0.2.3 (2016-11-20)

- correct page\_size value #216

### 0.2.2 (2016-10-31)

- add bans wave warning

### 0.2.1 (2016-10-03)

- fix tradepile/watchlist when consumable in pile (#194)
- fix card info url & bump default year in leagues/teams
- fix credits resetting to 0 on search (thanks to hunterjm #198)

### 0.2.0 (2016-09-26)

- fifa 17 & dump versions (thanks to rafaelget #192)

#### **0.1.10 (2016-04-30)**

- fix python 3 compatibility #183
- bump client version for and/ios (fix #190 thanks to rafaelget)
- bump user-agent and flash version

#### **0.1.9 (2015-12-11)**

- bump client version for and/ios
- bump user-agent \* flash version

#### **0.1.8 (2015-12-09)**

- core: fix #172, fix #176 crash when skuAccessList is empty

#### **0.1.7 (2015-11-30)**

- core: fix baseId calculation (thanks to hunterjm #174)

#### **0.1.6 (2015-11-19)**

- core: store credits after every call instead of making an additional call out

#### **0.1.5 (2015-11-15)**

- core: fix club (thanks to hunterjm #169)

#### **0.1.4 (2015-10-29)**

- core: fix itemParse (thanks to hunterjm #163)

#### **0.1.3 (2015-10-28)**

- core: bump clientversion for android/ios emulation
- core: add tradeStatus (thanks to hunterjm #161)
- exceptions: add code, reason, string to FutError

#### **0.1.2 (2015-09-28)**

- core: fix baseId calculation
- support app authentication (#147)

#### **0.1.1 (2015-09-19)**

- fix for ps/xbox

**0.1.0 (2015-09-17)**

- fifa 16
- core: update credits only on demand
- config: update user-agent (chrome 45 @ win10)

**0.0.24 (2015-02-11)**

- core: fix #135 type conversion in quickSell & watchlistDelete
- core: rename parameter squad\_num to squad\_id

**0.0.23 (2015-02-09)**

- urls: fix #131
- Captcha exception got img & token parameter
- core: add logout
- core: quickSell & watchlistDelete accepts now int/str or tuple/list with multiple ids
- urls: enable ssl for all urls
- core & urls: add timestamp dynamically (just right before every request)

**0.0.22 (2014-12-28)**

- setup: fix manifest
- core: save session if code is not provided but required

**0.0.21 (2014-12-13)**

- two-step verification
- fix cookies parameter not working (#99)
- core: use LWPCookieJar instead of pickle
- core: fix logging in \_\_sendToPile\_\_

**0.0.20 (2014-10-19)**

- fix typo

**0.0.19 (2014-10-19)**

- core: update old fut14 urls
- core: add cookies feature (save cookies after every request and load it when restarting app like browser)
- core: add saveSession, searchDefinition
- core: log sendToPile action

#### **0.0.18 (2014-10-01)**

- core: add methods to list and delete available messages (thanks to jamslater)
- core: rework base id from resource id calculation, use new constant (thanks to jamslater)
- core: update android \* ios clientVersion (9->11)

#### **0.0.17 (2014-09-22)**

- rename project (fut14->fut)
- fut15 (drop support for fifa 14)

#### **0.0.16 (2014-08-31)**

- #76 fix buying (thanks to arthurnn)

#### **0.0.15 (2014-08-29)**

- add new exceptions: doLoginFail, MaxSessions, Captcha
- add changelog
- NullHandler is default logging handler
- core: bump clientVersion (8->9)

#### **0.0.14 (2014-07-06)**

- core: relist returns number of delted/sold if clean parameter was set
- add new exception FeatureDisabled
- core: add emulate
- core: add stats
- core: add clubInfo

#### **0.0.13 (2014-04-19)**

- core: add sendToWatchlist

#### **0.0.12 (2014-02-23)**

- exceptions: add Unauthorized & MultipleSession
- fix quicksell

#### **0.0.11 (2014-02-15)**

- fix logger
- setup.py is now executable



#### 0.0.10 (2014-02-15)

- core: add clean ability to relist (remove sold cards)
- core: keepalive returns credit amount

#### 0.0.9 (2014-01-26)

- fix relist

#### 0.0.8 (2014-01-26)

- add new exception Conflict
- init docs
- core: add relist
- core: add sendToClub

#### 0.0.7 (2014-01-13)

- add few exceptions

#### 0.0.6 (2013-12-30)

- core: add DEBUG feature
- add multiplatform support (xbox/ps3/and/ios)

#### 0.0.5 (2013-12-23)

- core: add assetId param to searchAuction method
- core: add pileSize
- core: add leagueId to item data parser

#### 0.0.4 (2013-11-10)

- convert lowercase function/method names to mixedCase (send\_to\_tradepile -> sendToTradepile)
- drop python-2.5 (requests)
- core: python 3 support

#### 0.0.3 (2013-10-25)

- core: move requests session init & headers from login to init
- core: update credits on every request (only if it is available included in response)

### 0.0.2 (2013-10-17)

- core: add watchlist
- core: add card\_info function
- core: add alias for base\_id & card\_info

### 0.0.1 (2013-10-15)

- init

## 2.2 Installation

## 2.3 Introduction

### 2.3.1 Functionality

The fut library can perform 28 basic [REST](#) functions on your Fifa Ultimate Team. It also includes 6 properties that provide access to various FUT databases (nations, leagues, teams, stadiums, balls, players, playStyles).

---

### 2.3.2 The FUT Web App Structure

The basic layout of the FUT Web App is below. There are 5 primary categories: Squads, Squad Building Challenge, Transfers, Store, and Club. You can perform several actions in each category with the methods available in the fut library, but not all actions in the Web App have yet been mapped out. **Click on a category below** to learn more about the methods currently available in the fut library.

---

### 2.3.3 Other FUT databases

The fut library has the following databases inside of it. Click on the link next to each property below to view its contents.

- **players** [Contents](#) (*on Google Drive bc it is 18K rows*)
- **playStyles** [Contents](#)
- **nations** [Contents](#)
- **leagues** [Contents](#)
- **teams** [Contents](#) (*on Google Drive bc it is 900+ rows*)
- **stadiums** [Contents](#)

A database for consumables (contracts, healing, fitness, training, position changes, chemistry styles, managers) has not been included in the library, but a [cookbook recipe](#) to obtain detailed consumables information (catalogued by koolaidjones) can be found [here](#)..

## 2.4 Cookbook

### 2.4.1 Retrieve Non-Player Cards

The non-player cards are updated as of Fifa 2017 and are hosted as a csv on github. The function below uses the pandas library to provide a table of the non-player cards.

Retrieve Non-Player Cards

```
## Gets non-player card types
>>> import pandas as pd
>>> def nonPlayers():
    url = 'https://raw.githubusercontent.com/TrevorMcCormick/futmarket/master/
    ↪cardInfo.csv'
    return(pd.read_csv(url))
>>> nonPlayers().head()
```

	amount	assetid	subtypeid	pile	rareflag	rating	weightrare	year	\
0	0	7	201	7	0	50	0	2017	
1	0	7	201	7	0	65	0	2017	
2	0	7	201	7	0	80	0	2017	
3	0	7	201	7	1	60	100	2017	
4	0	7	201	7	1	70	100	2017	

	resourceid	bronze	silver	gold	class	category	level	type
0	5001001	8	2	1	Player	Contract	Bronze	Non-Rare
1	5001002	10	10	8	Player	Contract	Silver	Non-Rare
2	5001003	15	11	13	Player	Contract	Gold	Non-Rare
3	5001004	15	6	3	Player	Contract	Bronze	Rare
4	5001005	20	24	18	Player	Contract	Silver	Rare

## 2.5 IDs (object structures?)

List of ids is available below:

### 2.5.1 Consumable IDs

Consumable IDs have been added by [Koolaidjones](#). They are updated through 2017, so some IDs may be slightly off. The full table is available at this [Google Drive link](#).

### 2.5.2 League IDs

Leagues are found through the property `fut.leagues`.

League ID Lookup Table

ID	League
1	'Alka Superliga'
4	'Belgium Pro League'
7	'Liga do Brasil'
10	'Eredivisie'
13	'Premier League'
14	'EFL Championship'
16	'Ligue 1 Conforama'
17	'Domino's Ligue 2'
19	'Bundesliga'
20	'Bundesliga 2'
31	'Calcio A'
32	'Calcio B'
39	'Major League Soccer'
41	'Eliteserien'
50	'Scottish Premiership'
51	'Scotland League'
53	'LaLiga Santander'
54	'LaLiga 1 I 2 I 3'
56	'Allsvenskan'
57	'Colombia Apertura'
58	'Colombia Clausura'
60	'EFL League One'
61	'EFL League Two'
63	'Hellas Liga'
65	'SSE Airtricity League'
66	'Ekstraklasa'
67	'Russian Football Premier League'
68	'Süper Lig'
76	'Rest of World'
78	'Men's National'
80	'Österreichische Fußball-Bundesliga'
83	'K LEAGUE Classic'
84	'Mexican Clausura'
85	'Mexican Apertura'
152	'Torneo

de Primera' || 153 | 'Torneo de Primera' || 156 | 'Chile Apertura' || 157 | 'Chile Clausura' || 189 | 'Raiffeisen Super League' || 308 | 'Liga NOS' || 319 | 'Česká Liga' || 322 | 'Finnliiga' || 332 | 'Ukrayina Liha' || 335 | 'Campeonato Scotiabank' || 336 | 'Liga Dimayor' || 341 | 'LIGA Bancomer MX' || 347 | 'South African FL' || 349 | 'Meiji Yasuda J1 League' || 350 | 'Dawry Jameel' || 351 | 'Hyundai A-League' || 353 | 'Primera División' || 371 | 'Scotland League' || 382 | 'Free Agents' || 383 | 'Created Players League' || 384 | 'Creation Centre League' || 390 | 'MLS Cup' || 993 | 'Asia Qualifier' || 1003 | 'Copa Latinoamericana' || 1004 | 'Colombia Apertura' || 1005 | 'Colombia Finalización' || 1006 | 'Chile Apertura' || 1007 | 'Chile Clausura' || 1008 | 'Argentina Apertura' || 1009 | 'Argentina Clausura' || 2002 | 'Nacional B' || 2012 | 'China Top League' || 2025 | 'Liga do Brasil B' || 2028 | 'World League' || 2076 | '3. Liga' || 2096 | 'Special Teams' || 2118 | 'Icons' || 2136 | "Women's National" || 2138 | 'International Clubs' || 2150 | 'REWARDS' || 10001 | 'Denmark League 2' || 10004 | 'Belgium League 2' || 10007 | 'Liga do Brasil 1' || 10010 | 'Holland League 2' || 10017 | 'France League 3' || 10020 | 'Germany League 3' || 10032 | 'Italy League 3' || 10041 | 'Norway League 2' || 10050 | 'Scotland League 2' || 10054 | 'Spain League 3' || 10056 | 'Sweden League 2' || 10061 | 'England League 5' || 10065 | 'Ireland League 2' || 10066 | 'Poland League 2' || 10067 | 'Russia League 2' || 10076 | 'Rest of World 2' || 10080 | 'Austria League 2' || 10083 | 'Korea League 2' || 10189 | 'Switzerland League 2' || 10308 | 'Portugal League 2' || 10335 | 'Chile League 2' || 10336 | 'Colombia League 2' || 10341 | 'Mexico League 2' || 10350 | 'Saudi League 2' || 10353 | 'Argentina League 2' |

## 2.5.3 Player IDs

The player IDs are found through the property `fut.players`. A full table is available at this [Google Drive link](#).

## 2.5.4 Player Info Dict

The player info dict is returned by many functions. Below is an example with a helpful table of return types.

Player Info Dict Code Example

```
>>> #Get first player in my club
>>> fut.club()[0]
[{'assetId': 230621,
  'assists': 0,
  .....}]
```

Player Info Dict Lookup Table

field	type	description
assetId	int	unique asset id
assists	int	career assists
attributeList	dict	five primary stats
bidState	str	state of bid
buyNowPrice	int	coins to buy now
cardType	int	unsure
cardsubtypeid	int	unsure
contract	int	0-99 games
count	?	None
currentBid	int	coins of currentBid (0 if no bids)
discardValue	int	coins recieved from quick sell
expires	int	seconds until expires from transfer market
fitness	int	0-99 fitness
formation	str	current team formation
id	int	unique card id. one asset id can have many card ids (TOTW example)
injuryGames	int	games until current injury expires
injuryType	str	current injury type
itemState	str	what you can do with the current item
itemType	str	player, development, training
lastSalePrice	int	coins last sold for on transfer market
leagueId	int	use fut.leagues() to get dictionary
lifetimeAssists	int	career assists again
lifetimeStats	dict	all career stats
loyaltyBonus	int	unsure
morale	int	0-99 ... not sure what this does
nation	int	use fut.nations() to get dictionary
offers	int	number of bids in transfer market
owners	int	number of historical owners
pile	int	unsure
playStyle	int	use fut.playStyles() to get dictionary
position	str	preferred player position
rareflag	int	rare card
rating	int	0-99
resourceGameYear	int	2018
resourceId	int	same as assetid
sellerEstablished	int	unsure
sellerId	int	current seller on transfer market (empty)
sellerName	str	current seller on transfer market (empty)
startingBid	int	coins of the first bid on transfer market
statsList	dict	same as lifetimeStats
suspension	int	red card suspension games remaining
teamid	int	use fut.teams() to get dictionary
timestamp	int	epoch time that you acquired the item
tradeId	int	unique tradeId on transfer market
tradeState	str	current State on transfer market
training	int	unsure
untradeable	boolean	listable on the transfer market
untradeableCount	?	unsure
watched	boolean	currently in watchlist
year	int	2018

## 2.5.5 PlayStyle IDs

PlayStyle IDs are found through the property `fut.playstyles`.

PlayStyle ID Lookup Table

ID	Description
250	'BASIC'
251	'SNIPER'
252	'FINISHER'
253	'DEADEYE'
254	'MARKSMAN'
255	'HAWK'
256	'ARTIST'
257	'ARCHITECT'
258	'POWERHOUSE'
259	'MAESTRO'
260	'ENGINE'
261	'SENTINEL'
262	'GUARDIAN'
263	'GLADIATOR'
264	'BACKBONE'
265	'ANCHOR'
266	'HUNTER'
267	'CATALYST'
268	'SHADOW'
269	'WALL'
270	'SHIELD'
271	'CAT'
272	'GLOVE'
273	'GK BASIC'

## 2.5.6 Nation IDs

Nation IDs are found through the property `fut.nations`.

Nation ID Lookup Table

ID	Nation
1	'Albania'
2	'Andorra'
3	'Armenia'
4	'Austria'
5	'Azerbaijan'
6	'Belarus'
7	'Belgium'
8	'Bosnia Herzegovina'
9	'Bulgaria'
10	'Croatia'
11	'Cyprus'
12	'Czech Republic'
13	'Denmark'
14	'England'
15	'Montenegro'
16	'Faroe Islands'
17	'Finland'
18	'France'
19	'FYR Macedonia'
20	'Georgia'
21	'Germany'
22	'Greece'
23	'Hungary'
24	'Iceland'
25	'Republic of Ireland'
26	'Israel'
27	'Italy'
28	'Latvia'
29	'Liechtenstein'
30	'Lithuania'
31	'Luxemburg'
32	'Malta'
33	'Moldova'
34	'Netherlands'
35	'Northern Ireland'
36	'Norway'
37	'Poland'
38	'Portugal'
39	'Romania'
40	'Russia'
41	'San Marino'
42	'Scotland'
43	'Slovakia'
44	'Slovenia'
45	'Spain'
46	'Sweden'
47	'Switzerland'
48	'Turkey'
49	'Ukraine'
50	'Wales'
51	'Serbia'
52	'Argentina'
53	'Bolivia'
54	'Brazil'
55	'Chile'
56	'Colombia'
57	'Ecuador'
58	'Paraguay'
59	'Peru'
60	'Uruguay'
61	'Venezuela'
62	'Anguilla'
63	'Antigua & Barbuda'
64	'Aruba'
65	'Bahamas'
66	'Barbados'
67	'Belize'
68	'Bermuda'
69	'British Virgin Isles'
70	'Canada'
71	'Cayman Islands'
72	'Costa Rica'
73	'Cuba'
74	'Dominica'
75	'International'
76	'El Salvador'
77	'Grenada'
78	'Guatemala'
79	'Guyana'
80	'Haiti'
81	'Honduras'
82	'Jamaica'
83	'Mexico'
84	'Montserrat'
85	'Netherlands Antilles'
86	'Nicaragua'
87	'Panama'
88	'Puerto Rico'
89	'St Kitts Nevis'
90	'St Lucia'
91	'St Vincent Grenadine'
92	'Suriname'
93	'Trinidad & Tobago'
94	'Turks & Caicos'
95	'United States'
96	'US Virgin Islands'
97	'Algeria'
98	'Angola'
99	'Benin'
100	'Botswana'
101	'Burkina Faso'
102	'Burundi'
103	'Cameroon'
104	'Cape Verde Islands'
105	'CAR'
106	'Chad'
107	'Congo'
108	'Ivory Coast'
109	'Djibouti'
110	'DR Congo'
111	'Egypt'
112	'Equatorial Guinea'
113	'Eritrea'
114	'Ethiopia'
115	'Gabon'
116	'Gambia'
117	'Ghana'
118	'Guinea'
119	'Guinea Bissau'
120	'Kenya'
121	'Lesotho'
122	'Liberia'
123	'Libya'
124	'Madagascar'
125	'Malawi'
126	'Mali'
127	'Mauritania'
128	'Mauritius'
129	'Morocco'
130	'Mozambique'
131	'Namibia'
132	'Niger'
133	'Nigeria'
134	'Rwanda'
135	'São Tomé & Príncipe'
136	'Senegal'
137	'Seychelles'
138	'Sierra Leone'
139	'Somalia'
140	'South Africa'
141	'Sudan'
142	'Swaziland'
143	'Tanzania'
144	'Togo'
145	'Tunisia'
146	'Uganda'
147	'Zambia'
148	'Zimbabwe'
149	'Afghanistan'
150	'Bahrain'
151	'Bangladesh'
152	'Bhutan'
153	'Brunei Darussalam'
154	'Cambodia'
155	'China PR'
156	'Chinese Taipei'
157	'Guam'
158	'Hong Kong'
159	'India'
160	'Indonesia'
161	'Iran'
162	'Iraq'
163	'Japan'
164	'Jordan'
165	'Kazakhstan'
166	'Korea DPR'
167	'Korea Republic'
168	'Kuwait'
169	'Kyrgyzstan'
170	'Laos'
171	'Lebanon'
172	'Macau'
173	'Malaysia'
174	'Maldives'
175	'Mongolia'
176	'Myanmar'
177	'Nepal'
178	'Oman'
179	'Pakistan'
180	'Palestinian Authority'
181	'Philippines'
182	'Qatar'
183	'Saudi Arabia'
184	'Singapore'
185	'Sri Lanka'
186	'Syria'
187	'Tajikistan'
188	'Thailand'
189	'Turkmenistan'
190	'United Arab Emirates'
191	'Uzbekistan'
192	'Vietnam'
193	'Yemen'
194	'American Samoa'
195	'Australia'
196	'Cook Islands'
197	'Fiji'
198	'New Zealand'
199	'Papua New Guinea'
200	'Samoa'
201	'Solomon Islands'
202	'Tahiti'
203	'Tonga'
204	'Vanuatu'
205	'Gibraltar'
206	'Greenland'
207	'Dominican Republic'
208	'Estonia'
209	

‘Created Players’, || 210 | ‘Free Agents’, || 211 | ‘Rest of World’, || 212 | ‘Timor-Leste’, || 213 | ‘Chinese Taipei’, || 214 | ‘Comoros’, || 215 | ‘New Caledonia’, || 219 | ‘Kosovo’ |

## 2.5.7 Rare Cards

There are currently 41 confirmed types of rare cards.

Rare Card Lookup Table

| Description | ID | |———| | | NONE | 0 | | RARE | 1 | | LOCK | 2 | | TOTW | 3 | | PURPLE | 4 | | TOTY | 5 | | RB | 6 | | GREEN | 7 | | ORANGE | 8 | | PINK | 9 | | TEAL | 10 | | TOTS | 11 | | LEGEND | 12 | | WC | 13 | | UNICEF | 14 | | OLDMOTM | 15 | | FUTTY | 16 | | STORYMODE | 17 | | CHAMPION | 18 | | CMOTM | 19 | | IMOTM | 20 | | OTW | 21 | | HALLOWEEN | 22 | | MOVEMBER | 23 | | SBC | 24 | | SBCP | 25 | | PROMOA | 26 | | PROMOB | 27 | | AWARD | 28 | | BDAY | 30 | | UNITED | 31 | | FUTMAS | 32 | | RTRC | 33 | | PTGS | 34 | | FOF | 35 | | MARQUEE | 36 | | CHAMPIONSHIP | 37 | | EUMOTM | 38 | | TOTT | 39 | | RRC | 40 | | RRR | 41 |

## 2.5.8 Stadium IDs

Stadiums IDs are found through the property fut.stadiums.

Stadium ID Lookup Table

| ID | Stadium | |———| | 1 | ‘Old Trafford’ | | 2 | ‘Santiago Bernabéu’ | | 4 | ‘Stade Gerland’ | | 5 | ‘San Siro’ | | 6 | ‘Camp Nou’ | | 8 | ‘Stadio Delle Alpi’ | | 9 | ‘Signal Iduna Park’ | | 10 | ‘Estadio Mestalla’ | | 13 | ‘Anfield’ | | 14 | ‘Parc des Princes’ | | 15 | ‘Amsterdam ArenA’ | | 16 | ‘Stade Felix Bollaert’ | | 17 | ‘Constant Vanden Stock’ | | 25 | ‘Closed Square Style’ | | 26 | ‘Forest Park Stadium’ | | 28 | ‘Stamford Bridge’ | | 29 | ‘Orange Vélodrome’ | | 30 | ‘Veltins Arena’ | | 32 | ‘Crown Lane’ | | 33 | ‘Union Park Stadium’ | | 34 | ‘Town Park’ | | 35 | ‘Euro Park’ | | 37 | ‘Div 3 Euro Style’ | | 38 | ‘Urban Training’ | | 39 | ‘Rural Training’ | | 41 | ‘Volksparkstadion’ | | 42 | ‘Estadio Vicente Calderón’ | | 100 | ‘St. James’ Park’ | | 102 | ‘José Alvalade’ | | 104 | ‘Estadio Azteca’ | | 107 | ‘Estádio da Luz’ | | 108 | ‘Seoul Sang-am Stadium’ | | 110 | ‘Daegu Stadium’ | | 111 | ‘Estádio do Dragão’ | | 112 | ‘Fratton Park’ | | 113 | ‘St. Mary’s Stadium’ | | 115 | ‘Villa Park’ | | 116 | ‘White Hart Lane’ | | 124 | ‘Small Olympic’ | | 127 | ‘Large Olympic’ | | 129 | ‘Large Square’ | | 133 | ‘Mercedes-Benz Arena’ | | 134 | ‘HDI Arena’ | | 135 | ‘Olympiastadion’ | | 137 | ‘Allianz Arena’ | | 138 | ‘Commerzbank Arena’ | | 147 | ‘Stadion Europa’ | | 149 | ‘Al Jayeed Stadium’ | | 153 | ‘Aloha Park’ | | 155 | ‘Wembley Stadium’ | | 345 | ‘King Abdullah Sports City’ | | 156 | ‘Emirates Stadium’ | | 157 | ‘Stadio Olimpico’ | | 158 | ‘Estadio de las Artes’ | | 161 | ‘StubHub Center’ | | 163 | ‘Jalisco’ | | 165 | ‘Stade de Suisse’ | | 172 | ‘StadiumName\_172\_FIWC-Stadium\_FullChar’ | | 174 | ‘Stadio Comunale’ | | 175 | ‘Arena del Centenario’ | | 176 | ‘Waldstadion’ | | 177 | ‘La Canchita’ | | 178 | ‘Stadion Neder’ | | 179 | ‘Stade Municipal’ | | 180 | ‘Ivy Lane’ | | 181 | ‘El Grandioso’ | | 182 | ‘Stadion 23. Maj’ | | 183 | ‘Estadio El Medio’ | | 184 | ‘North America’ | | 185 | ‘South America’ | | 186 | ‘Southern Europe’ | | 187 | ‘Eastern Europe’ | | 188 | ‘Central Europe’ | | 189 | ‘United Kingdom’ | | 190 | ‘Asia’ | | 192 | ‘El Libertador’ | | 193 | ‘Stadio Classico’ | | 194 | ‘Eastpoint Arena’ | | 195 | ‘Stadion Olympik’ | | 196 | ‘Stadion Hanguk’ | | 197 | ‘O Dromo’ | | 211 | ‘Estadio Chamartin’ | | 212 | ‘Estadio Presidente G.Lopes’ | | 213 | ‘Green Point Stadium’ | | 214 | ‘Durban Stadium’ | | 215 | ‘Ellis Park Stadium’ | | 216 | ‘Soccer City Stadium’ | | 217 | ‘Free State Stadium’ | | 218 | ‘Nelson Mandela Bay Stadium’ | | 219 | ‘Mbombela Stadium’ | | 220 | ‘Peter Mokaba Stadium’ | | 221 | ‘Royal Bafokeng Stadium’ | | 222 | ‘Loftus Versfeld Stadium’ | | 223 | ‘Friður Stadium’ | | 224 | ‘Satta Stadium’ | | 225 | ‘Akaaroa Stadium’ | | 226 | ‘Hasiti Arena’ | | 227 | ‘Salam Stadium’ | | 228 | ‘Court Lane’ | | 229 | ‘Arena D’Oro’ | | 233 | ‘Peuan Arena’ | | 234 | ‘Pyonghwa Stadium’ | | 235 | ‘Udugu Stadium’ | | 236 | ‘El Coloso’ | | 238 | ‘Africa’ | | 246 | ‘Etihad Stadium’ | | 247 | ‘Allianz Stadium’ | | 248 | ‘BC Place Stadium’ | | 249 | ‘Molton Road’ | | 250 | ‘Oceanic Arena’ | | 253 | ‘Olympic Stadium’ | | 254 | ‘Municipal Stadium Poznan’ | | 255 | ‘National Stadium Warsaw’ | | 256 | ‘Arena Gdansk’ | | 257 | ‘Municipal Stadium Wroclaw’ | | 258 | ‘Metalist Stadium’ | | 259 | ‘Arena Lviv’ | | 260 | ‘Donbass Arena’ | | 261 | ‘Sanderson Park’ | | 262 | ‘King Fahd Stadium’ | | 264 | ‘La Bombonera’ | | 265 | ‘Goodison Park’ | | 267 | ‘Arena Amazonia’ | | 268 | ‘Arena da Baixada’ | | 269 | ‘Arena de Sao Paulo’ | | 270 | ‘Arena Fonte Nova’ | | 271 | ‘Arena Pantanal’ | | 272 | ‘Arena Pernambuco’ | | 273 | ‘Estadio Beira-Rio’ | | 274 | ‘Estadio Castela’ | | 275 | ‘Estadio das Dunas’ | | 276 | ‘Estadio do Maracana’ | | 277 | ‘Estadio Mineirao’ | | 278 | ‘Estadio Nacional’ | | 279 | ‘Baba Yetu Stadium’ | | 282 | ‘Stade du 13 Octobre’ | | 285 | ‘Stade du Lukanga’ | | 286 | ‘Estadio de

las Cascadas' || 287 | 'El Grandioso de las Pampas' || 288 | 'Singeom Stadium' || 289 | 'Shibusaka Stadium' || 290 | 'Gold Lake Stadium' || 291 | 'Stadio San Dalla Pace' || 316 | 'Training Centre' || 325 | 'Boleyn Ground' || 326 | 'Stadium of Light' || 327 | 'The Hawthorns' || 329 | 'Carrow Road' || 330 | 'Selhurst Park' || 331 | 'Stoke City FC Stadium' || 332 | 'KCOM Stadium' || 333 | 'Liberty Stadium' || 335 | 'King Power Stadium' || 336 | 'Turf Moor' || 337 | 'Loftus Road' || 340 | 'The Amex Stadium' || 341 | 'CenturyLink Field' || 343 | 'BORUSSIA-PARK' || 344 | 'El Monumental' || 345 | 'King Abdullah Sports City' || 347 | 'Vicarage Road' || 348 | 'Vitality Stadium' || 349 | 'Riverside Stadium' || 353 | 'Training Ground' || 354 | 'Suita City Football Stadium' || 355 | 'London Stadium' || 358 | 'EA SPORTS FIFA Stadium' || 364 | 'Wanda Metropolitano' || 372 | 'Luzhniki Stadium' || 373 | 'Saint Petersburg Stadium' || 374 | 'Kazan Arena' || 375 | 'Samara Arena' || 376 | 'Fisht Stadium' || 377 | 'Nizhny Novgorod Stadium' || 378 | 'Spartak Stadium' || 379 | 'Rostov Arena' || 380 | 'Ekaterinburg Arena' || 381 | 'Mordovia Arena' || 382 | 'Volgograd Arena' || 383 | 'Kaliningrad Stadium' || 384 | 'Kirklees Stadium' |

## 2.5.9 Team IDs

The team IDs are found through the property `fut.teams`. A full table is available at this [Google Drive link](#).

### 2.5.10 Trade Status Dict

The trade status dict returns basic info about a tradeId.

Trade Status Dict Code Example

```
>>> fut.tradeStatus(16575379694)
[{'tradeId': 16575379694,
  'buyNowPrice': 1800,
  'tradeState': 'closed',
  ...}]
```

Trade Status Dict Lookup Table

field	type	description	tradeId	int	NA	buNowPrice	int	NA	tradeState	str	NA	bidState	str	NA	startingBid	int	NA	id	int	NA	offers	int	NA	currentBid	int	NA	expires	int	NA	sellerEstablished	int	NA	sellerId	int	NA	sellerName	str	NA	watched	boolean	NA	resourceId	int	NA	discardValue	int	NA
-------	------	-------------	---------	-----	----	------------	-----	----	------------	-----	----	----------	-----	----	-------------	-----	----	----	-----	----	--------	-----	----	------------	-----	----	---------	-----	----	-------------------	-----	----	----------	-----	----	------------	-----	----	---------	---------	----	------------	-----	----	--------------	-----	----

### 2.5.11 Transfer Info Dict

The transfer info dict is returned by the `tradepile` and `watchlist` functions. Below is an example with a helpful table of return types.

Transfer Info Dict Code Example

```
>>> #Get first player in my transfers
>>> fut.tradepile()[0]
{'tradeId': 16575379694, 'buyNowPrice': 1800, 'tradeState': 'closed'....}
```

Transfer Info Dict Lookup Table

field	type	description	tradeId	int	NA	buNowPrice	int	NA	tradeState	str	NA	bidState	str	NA	startingBid	int	NA	id	int	NA	offers	int	NA	currentBid	int	NA	expires	int	NA	sellerEstablished	int	NA	sellerId	int	NA	sellerName	str	NA	watched	boolean	NA	resourceId	int	NA	discardValue	int	NA	timestamp	int	NA	rating	int	NA	assetId	int	NA	itemState	str	NA	rareflag	int	NA	formation	str	NA	leagueId	int	NA	injuryType	str	NA	injuryGames	int	NA	lastSalePrice	int	NA	fitness	int	NA	training	int	NA	suspension	int	NA
-------	------	-------------	---------	-----	----	------------	-----	----	------------	-----	----	----------	-----	----	-------------	-----	----	----	-----	----	--------	-----	----	------------	-----	----	---------	-----	----	-------------------	-----	----	----------	-----	----	------------	-----	----	---------	---------	----	------------	-----	----	--------------	-----	----	-----------	-----	----	--------	-----	----	---------	-----	----	-----------	-----	----	----------	-----	----	-----------	-----	----	----------	-----	----	------------	-----	----	-------------	-----	----	---------------	-----	----	---------	-----	----	----------	-----	----	------------	-----	----

```
contract | int | NA || position | str | NA || playStyle | int | NA || itemType | str | NA || cardType | it | NA || cardsubtypeid  
| int | NA || owners | int | NA || untradeable | boolean | NA || morale | int | NA || statsList | list | NA || lifetimeStats | list  
| NA || attributeList | list(dict) | NA || teamid | int | NA || assists | int | NA || lifetimeAssitss | int | NA || loyaltyBonus  
| int | NA || pile | int | NA || nation | int | NA || year | int | NA || resourceGameYear | int | NA || marketDataMinPrice |  
int | NA || marketDataMaxPrice | int | NA || loans | int | NA |
```



In this part You can find all methods descriptions.

## 3.1 Squads

Below is the current state of functionality within the **Squads** category.

One method exists to return the players on your active squad, without managers or other items. Squad management and TOTW are not included yet in the fut library.

---

### 3.1.1 fut.squad()

arguments: (self, squad\_id=0, persona\_id=None)

fut.squad() returns a list of dictionaries, each containing information about one player in your active squad. *The **player info dict** is linked here.*

*Example:*

```
>>> len(fut.squad())
23
>>> fut.squad()
[{'assetId': 230621,
  'assists': 0,
  'attributeList': [{u'index': 0, u'value': 88},
                    {u'index': 1, u'value': 78},
                    {u'index': 2, u'value': 72},
                    {u'index': 3, u'value': 88},
                    {u'index': 4, u'value': 46},
                    {u'index': 5, u'value': 78}],
  'bidState': None,
```

```
'buyNowPrice': None,
.....}]
```

## 3.2 Squad Building Challenge

Below is the current state of functionality within the **Squad Building Challenge** category. In the `core.py` file, three more methods exist (`sbsSetChallenges`, `sbsSquad`, and `sendToSbs`) but do not currently work.

There is one working method in this category that GETs Squad Building Challenge info.

---

### 3.2.1 fut.sbsSets()

`fut.sbsSets()` returns a dictionary of a list of dictionaries of Squad Building Challenge categories, each containing active challenges with descriptions and other info.

```
>>> ## General function
>>> fut.sbsSets()
{'categories': [{u'categoryId': 8,
  u'name': u'BLACK FRIDAY',
  u'priority': 1,
  u'sets': [{u'awards': [{u'count': 1,
  .....}
>>> ## What can we pull out of the dictionary?
>>> fut.sbsSets().keys()
[u'categories']
>>> ## What are my SB challenge categories called and how many challenges are within_
↳them?
>>> for c in fut.sbsSets()['categories']:
>>>     print c['name'], len(c['sets'])
ICONS 14
LEAGUES 6
ADVANCED 3
LIVE 4
UPGRADES 3
BASIC 3
>>> ## Give me IDs and names of all the challenges in the BASIC category
>>> # Make a generator-- this can only be used once so re-run the line below each_
↳time you use the generator
>>> basic_sbs = (s for s in fut.sbsSets()['categories'] if s['name'] == 'BASIC')
>>> for s in icons_sets:
        for c in s['sets']:
            print c['setId'], c['description']
1 Let's Get Started
2 League and Nation Basics
10 Let's Keep Going
```

## 3.3 Transfers

Below is the current state of functionality within the **Transfers** category. All methods within the Transfers category are stable.

---

The Transfers category contains many functions. We'll go through them in three sections: Search, Transfer List, and Transfer Targets.

## 3.4 Search

There are two functions in the search category: `search()` and `searchDefinition()`.

### 3.4.1 fut.search()

`fut.search()` returns a list of dictionaries that include the information for players on the transfer market. These are returned in ascending order of seconds until expiration. just like the Web App and the console experience. [A description of the returned dict of player info is linked here.](#)

There are many arguments available to filter your search request:

Search Arguments Table

argument	type	description	ctype
card type (player, development, training)	str	card level (bronze, silver, gold)	str
level	str	card category (fitness, health, etc.)	str
assetId	int	unique player id	int
defId	int	each assetId can have multiple defIds (ex. TOTW player card)	int
min_price	int	minimum currentBid	int
max_price	int	maximum currentBid	int
min_buy	int	minimum buyNow	int
max_buy	int	maximum buyNow	int
league	int	leagueId (available in fut.leagues)	int
club	int	clubId (available in fut.teams)	int
position	str	player preferred position abbreviation	str
rare	boolean	TRUE if rare card	boolean
playStyle	int	playStyleId (available in fut.playStyles)	int
start	int	page to start on (indexed at 0) through the web app.	int
page_size	str	cards to show on one page (range between 16-50)	str

*Example:*

```
>>> fut.search(ctype='player', level = 'gold')
[{'assetId': 230621,
  'assists': 0,
  'attributeList': [{u'index': 0, u'value': 88},
                    {u'index': 1, u'value': 78},
                    {u'index': 2, u'value': 72},
                    {u'index': 3, u'value': 88},
                    {u'index': 4, u'value': 46},
                    {u'index': 5, u'value': 78}],
  'bidState': None,
  'buyNowPrice': None,
  .....}]
```

### 3.4.2 fut.searchDefinition()

`fut.searchDefinition()` takes one argument (*it actually takes 3 but you only need one*), `assetId`, and it returns a list of dictionaries that include the information for specific variations of player cards by `assetId`. [A description of the returned dict of player info is linked here.](#)

*Example:*

```
>>> # Get variations for Mats Hummels
>>> fut.searchDefinition(item_id=178603)
[{'tradeId': None,
 'buyNowPrice': None,
 'tradeState': None
 ...}]
```

These are returned in descending order of rating. Card IDs are distinguished by the value of the *rarecard* field. [A table of card IDs are here.](#)

---

## 3.5 Transfer List

### 3.5.1 fut.tradepile()

`fut.tradepile()` returns a list of dictionaries that include the transfer information for players you’ve listed on the transfer market. [A description of the returned dict of transfer info is linked here.](#)

*Example:*

```
>>> fut.tradepile()
[{'tradeId': 16575379694,
 'buyNowPrice': 1800,
 'tradeState': 'closed'...}]
```

### 3.5.2 fut.tradeStatus()

`fut.tradeStatus()` takes one argument (`trade_id`) and returns a list containing a condensed dictionary for each `tradeId`. [The returned dictionary is linked here.](#)

*Example:*

```
>>> fut.tradeStatus(trade_id=16575379694)
[{'tradeId': 16575379694,
 'buyNowPrice': 1800,
 'tradeState': 'closed',
 ...}]
```

### 3.5.3 fut.sendToTradepile()

`fut.sendToTradepile()` takes one argument (`item_id`) and has an optional argument (`safe`) that checks the length of your tradepile to make sure you have room to store another item. The `item_id` argument is the `id` field in [player info dictionaries](#). A successful send will return `True`. An unsuccessful send will return `False` if you do not own the item you’re trying to send to the tradepile, or `403` if the item you’re trying to send is untradeable.

*Examples:*

```
>>> # Card I own where untradeable == False
>>> fut.sendToTradepile(item_id=117860780888)
True

>>> # Card I don't own
```

```
>>> fut.sendToTradepile(item_id=1)
False

>> # Card I own where untradeable == True
>> fut.sendToTradepile(item_id=118360247419)
{'Access-Control-Expose-Headers': 'Retry-After', 'Content-Length': '20', 'X-
↳UnzippedLength': '0', 'Content-Encoding': 'gzip', 'Server': 'Jetty(8.0.0.M2)'}
403
...}
```

### 3.5.4 fut.tradepileDelete()

`fut.tradepileDelete()` takes one argument (`item_id`). The `item_id` argument is the `id` field in [player info dictionaries](#). A successful delete will return `True`. An unsuccessful delete will return `fut.exceptions.Conflict` if the item you're trying to delete is in an active tradestate. You will receive a 410 error if you do not own the item you're trying to delete.

*Examples:*

```
>>> # Card in the tradepile that is not active
>>> fut.tradepileDelete(item_id=16575379694)
True

>>> # Card in the tradepile that is active.
>>> fut.tradepileDelete(item_id=16705837956)
fut.exceptions.Conflict

>> # Card I don't own
>> fut.tradepileDelete(item_id=1)
{'Content-Length': '20', 'X-UnzippedLength': '0', 'Content-Encoding': 'gzip', 'Access-
↳Control-Expose-Headers': 'Retry-After', 'Server': 'Jetty(8.0.0.M2)'}
410
...}
```

### 3.5.5 fut.tradepileClear()

`fut.tradepileClear()` takes zero arguments. It clears the `sold` items from your tradepile. It does not return anything for a successful clear, but it does return a 410 error if you don't have any `sold` items to clear from your tradepile.

```
>>> # Check number of cards in tradepile
>>> len(fut.tradepile())
3
>>> fut.tradepileClear()
>>> len(fut.tradepile())
2

>>> # Already cleared sold cards. Trying to clear again...
>>> fut.tradepileClear()
{'Content-Length': '20', 'X-UnzippedLength': '0', 'Content-Encoding': 'gzip', 'Access-
↳Control-Expose-Headers': 'Retry-After', 'Server': 'Jetty(8.0.0.M2)'}
410
```

### 3.5.6 fut.relist()

`fut.relist()` takes zero arguments. It relists the cards in your tradepile with the previous transfer parameters (`startingBid`, `buyNow`, `duration`). The function returns a dictionary with one key: `tradeIdList`. This contains a list of the `tradeIds` that were successfully relisted.

```
>>> fut.relist()
{'tradeIdList': [16575379694]}
```

### 3.5.7 fut.sell()

`fut.sell()` takes five arguments:

- `item_id` (int): the `id` field in [player info dictionaries](#)
- `bid` (int): the amount of coins you're willing to place the specific item/player up for starting bid (*`marketDataMinPrice` < `bid` < `marketDataMaxPrice`*)
- `buy_now` (int): the amount of coins you're willing to place the specific item/player up to buy now (*this must be higher than the starting bid*)
- `duration` (int): the amount of seconds (default 3600) to place the item/player up for transfer. (*this must be in intervals available through the web app*)
- `fast` (boolean): (default False) check the trade status of the item before listing on the transfer market

A successful listing will return the `tradeId` for your item.

```
>>> # Successful listing of an item I own listed within parameters
>>> fut.sell(item_id=119119825851, bid=1000, buy_now=100000, duration=3600)
16895235756

>>> # Unsuccessful listing of an item because I don't own it
>>> fut.sell(item_id=2, bid=1000, buy_now=10000)
PermissionDenied: 461

>>> # Unsuccessful listing of an item because it is listed below marketDataMinPrice
>>> fut.sell(item_id=119119825851, bid=200, buy_now=10000)
PermissionDenied: 461

>>> # Unsuccessful listing of an item because it is listed at a bad duration
>>> fut.sell(item_id=119119825851, bid=1000, buy_now=100000, duration=3599)
PermissionDenied: 460
```

---

## 3.6 Watch List

### 3.6.1 fut.watchlist()

`fut.watchlist()` returns a list of dictionaries that include the transfer information for players you've bid on on the transfer market. [A description of the returned dict of transfer info is linked here.](#)

*Example:*

```
>>> fut.watchlist()
[{'tradeId': 16656454826,
 'buyNowPrice': 5900000,
 'tradeState': 'active'...
```

### 3.6.2 fut.sendToWatchlist()

`fut.sendToWatchlist()` takes one argument (`trade_id`). The `trade_id` argument is the `tradeId` field in the [transfer info dictionary](#). A successful send will return an empty dictionary (*needs to be fixed to return something*). An unsuccessful send will return `fut.exceptions.Conflict` if the card has expired from the transfer market, or `fut.exceptions.PermissionDenied: 461` if the `id` you've provided is not valid.

*Examples:*

```
>>> # Active card on transfer market
>>> fut.sendToWatchlist(trade_id=117860780888)
{}

>> # Card that has expired or closed from transfer market
>> fut.sendToWatchlist(trade_id=118360247419)
fut.exceptions.Conflict

>>> # Ineligible card
>>> fut.sendToWatchlist(trade_id=2)
fut.exceptions.PermissionDenied: 461
```

### 3.6.3 fut.watchlistDelete()

`fut.watchlistDelete()` takes one argument (`trade_id`). The `trade_id` argument is the `tradeid` field in the [transfer info dictionary](#). A successful delete will return `True`. An unsuccessful delete will still return `True` (*needs to be updated*).

*Examples:*

```
>>> # Card in my watchlist
>>> fut.watchlistDelete(16746617493)
True

>>> # Card not in my watchlist
>>> fut.tradepileDelete(1)
True
```

### 3.6.4 fut.bid()

`fut.bid()` takes three arguments:

- `trade_id` (int): the `tradeId` field in [transfer info dictionaries](#)
- `bid` (int): the amount of coins you're willing to bid on the specific item/player (*must be higher than currentBid and startingBid but there is no maximum value*)
- `fast` (boolean): (default `False`) checks `tradeStatus` of item and your current coin count. this runs about 5x faster in the example for me below.

A successful bid will return True, and this player/item will be available in your watchlist. An unsuccessful bid could happen for a couple of reasons: either the transfer listing has expired or someone has outbid you.

```
>>> # Successful bid
>>> fut.sell(trade_id=16894052507, bid=150)
True

>>> # Unsuccessful bid because I bid less than the minimum amount
>>> fut.sell(trade_id=16894052507, bid=100)
False

>>> # Unsuccessful bid because the item expired or someone outbid me
>>> fut.sell(trade_id=16894049525, bid=200)
False

>>> # Checking how fast the fast-bid really is
>>> # fast=True
>>> start_time = time.time()
>>> fut.bid(trade_id=16894205969, bid=150, fast=True)
True
>>> elapsed_time = time.time() - start_time
>>> print(elapsed_time)
0.5593163967132568
>>> # fast=False
>>> start_time = time.time()
>>> fut.bid(trade_id=16894232971, bid=150, fast=False)
>>> elapsed_time = time.time() - start_time
>>> print(elapsed_time)
2.7050693035125732
```

### 3.6.5 fut.unassigned()

fut.unassigned() takes zero arguments. It provides player or item info for the unassigned items in your watchlist. These are typically items for which you paid the buyNow price. It returns the [player info dictionary](#).

```
>>> # Unassigned items
>>> fut.unassigned()
[{'assetId': 230621,
  'assists': 0,
  'attributeList': [{u'index': 0, u'value': 88}
  ...]}
>>> # Nothing unassigned
>>> fut.unassigned()
[]
```

### 3.6.6 fut.sendToClub()

fut.sendToClub() takes one argument, item\_id. The item\_id argument is the id field in [player info dictionaries](#). A successful send will return True, and an unsuccessful send will return False.

```
>>> # Item I own and want to send to my club
>>> fut.sendToClub(item_id=119293105688)
True
```



```
>>> # Item I do not own
>>> fut.sendToClub(item_id=2)
False
```

## 3.7 Store

Below is the current state of functionality within the **Store** category.

Two methods exist in the Store category

---

### 3.7.1 fut.buyPack()

fut.buyPack() takes two arguments: pack\_id and currency (default: 'COINS'). We're still working on a list of pack\_ids. Until then, this function is lacking documentation.

### 3.7.2 fut.openPack()

fut.openPack() takes one argument: pack\_id. We're still working on a list of pack\_ids. Until then, this function is lacking documentation.

## 3.8 Club

Below is the current state of functionality within the **Club** category.

Eight methods exist in the Club category.

---

### 3.8.1 fut.club()

fut.club() returns a list of dictionaries that include the information for players in your club. [A description of the returned dict of player info is linked here.](#)

There are many arguments available to filter your club search request:

Club Arguments Table

argument	type	description	ctype
card type	(player, development, training)	(default: player)	defId   int   unique card id. one asset id can have many card ids (TOTW example)
start	int	start page on WebApp ( <i>needs clarification</i> )	count   int   number of cards to return on one page (default 91)
level	str	card level (bronze, silver, gold)	category   str   card category (fitness, health, etc.)
assetId	int	unique player id	league   int   leagueId (available in fut.leagues)
club	int	clubId (available in fut.teams)	position   str   player preferred position abbreviation
zone	?	nationality   int   nationalityId (available in fut.nations)	rare   boolean   default False
playStyle	id	playStyleId (available in fut.playStyles)	

*Example:*

```
>>> fut.club()
[{'assetId': 230621,
  'assists': 1,
  'attributeList': [{u'index': 0, u'value': 88},
    {u'index': 1, u'value': 78},
    {u'index': 2, u'value': 72},
    {u'index': 3, u'value': 88},
    {u'index': 4, u'value': 46},
    {u'index': 5, u'value': 78}],
  .....}]
```

### 3.8.2 fut.clubStaff()

fut.clubStaff() returns a dictionary of a list of dictionaries that includes the bonuses you receive from specific staff in your club. [This will be documented later, but here is a helpful guide to Fifa 18 Staff cards.](#)

*Example:*

```
>>> fut.clubStaff()
{'bonus': [{'type': 'dribbling', 'value': 10},
  {'type': 'fitness', 'value': 5},
  {'type': 'gkDiving', 'value': 15},
  {'type': 'contract', 'value': 3},
  {'type': 'managerTalk', 'value': 0},
  {'type': 'physioArm', 'value': 5},
  {'type': 'physioFoot', 'value': 15},
  {'type': 'physioHip', 'value': 5}]}
```

### 3.8.3 fut.clubConsumables()

fut.clubConsumables() returns a list of dictionaries that includes the consumable cards and their details in your club. [A table of all 2017 consumable IDs and information is available here.](#) *2018 IDs have not yet been confirmed.*

*Example:*

```
>>> fut.clubConsumables()
[{'bidState': None,
  'bronze': 15,
  'buyNowPrice': None,
  'cardassetid': 7,
  'consumables': None,
  'consumablesContractManager': None,
  'consumablesContractPlayer': None,
  ...}]
```

### 3.8.4 fut.quickSell

fut.quickSell() discards an item (id) in your club for its discardValue. It returns True on a successful quickSell and returns UnknownError: b' ' for an unsuccessful quickSell. Untradeable cards can be discarded even though their discardValue is 0.

*Example:*

```
>>> # Card I own where untradeable == False
>>> fut.quickSell(item_id = 118917563073)
True

>>> # Card I do not own
>>> fut.quickSell(item_id = 2)
UnknownError: b''
```

### 3.8.5 fut.applyConsumable

`fut.applyConsumable()` takes two arguments: a consumableId (resourceId) that you own, and a player item\_id (id) that you own. It doesn't return anything for a successful consumable application, but it returns `UnknownError: b''` for an unsuccessful request. *There is currently not a method to apply a team consumable.*

*Example:*

```
>>> # Card and Consumable I own
>>> fut.applyConsumable(item_id = 119175722619, resource_id = 5001003)

>>> # Card and Consumable I do not own
>>> fut.applyConsumable(item_id = 119175722619, resource_id = 2)
UnknownError: b''
```

### 3.8.6 fut.keepalive()

`fut.keepalive()` is a simple function that returns your coin count. It is also a useful API call that tells the Fifa Web App that your session is still active.

*Example:*

```
>>> # Return my coin count
>>> fut.keepalive()
1002231023 #just kidding!
15598
```

### 3.8.7 fut.messages()

`fut.messages()` returns any active messages if you have them. I don't have any examples of this but any messages that include new Kits would be in here.

### 3.8.8 fut.objectives()

`fut.objectives()` returns a list of dictionaries containing your daily and weekly objectives.

*Example:*

```
>>> fut.objectives()
{'coinsAutoClaimed': 0,
 'dailyObjectives': [{'currentProgress': 0,
 'description': 'Get on the pitch today and play a game in Online Seasons mode',
 'difficulty': 3,
```

```
'expiryTime': 1513414800
...}},
'dailyRewardsAutoClaimed': False,
'itemsAutoClaimed': 0,
'packsAutoClaimed': 0,
'weeklyObjectives': [{'currentProgress': 0,
  'description': 'Win three or more Squad Battles matches this week to earn the_
↪FUTmas Elf kit [Untradeable]',
  'difficulty': 39,
  'expiryTime': 1513965600
...}},
'weeklyRewardsAutoClaimed': False}
>>> # Let's say I want to see the names of all of my daily objectives
>>> for i in fut.objectives().get('dailyObjectives'):
>>>     print(i['name'])
Becoming Seasoned
Buy a Midfielder
Use your Head
Four the Bundesliga
Get Fit
```

If you want to contribute to the project, this part of the documentation is for you.

### 4.1 Authors

Fut is written and maintained by Piotr Staroszczyk and various contributors:

#### 4.1.1 Development Lead

- Piotr Staroszczyk <[piotr.staroszczyk@get24.org](mailto:piotr.staroszczyk@get24.org)>

#### Documentation maintainer

- Trevor McCormick @TrevorMcCormick

#### 4.1.2 EAHashingAlgorithm

- Danny Cullen @dcullen88

#### 4.1.3 Patches and Suggestions

- mvillarejo
- Mauro Marano
- Innursery
- Arthur Nogueira Neves @arthurnn
- jamslater

- [rjansen](#)
- [ricklhp7](#)
- [hunterjm](#)
- [fifa2017player](#)
- [bas85](#)
- [spacedlevo](#)
- [pulkitsharma](#)
- [xAranaktu](#)
- [LasseRegin](#)
- [kirov](#)
- [jsarasti](#)
- [Trevor McCormick @TrevorMcCormick](#)