

**Table of Contents**

API Manual

21 March 2020

Introduction 4

Requests and Responses 5

Authentication and Authorization 7

Implementation tricks 7

Getting started using API 8

L18N and I13n 8

User-Agent 8

Synchronization and performance 8

Intro to data reference 9

Query format 9

General Features 10

API is Alive 11

Shooters 11

Sign-up or bulk-creating users/shooters 11

General information for myself 11

Change password 12

Verifying score using PIN or password 12

Model, fields, extras and excluded 12

Armoury 21

URLs 21

LogEntry 21

Exercise 29

Goals 29

Firearm 30

Sight 46

Ammunition 49

Handload Recipe 52

IPSC/USPSA Matches 55

URLs 55

Creating or updating score sheets 56

Model, fields, extras and excluded 57

IPSC/USPSA Series 65

URLs 65

IDPA Matches 66

URLs 66

Creating or updating score sheets 66

Model, fields, extras and excluded 67

PPC Matches 71

URLs 71

Creating or updating score sheets 71

Model, fields, extras and excluded 72

PPC Series 76

URLs 76

Steel Matches 77

URLs 77

Creating or updating score sheets 77

Model, fields, extras and excluded 77

SASS / CAS Matches 80

URLs 80

Creating or updating score sheets 80

Model, fields, extras and excluded 81

NORDIC Matches 82

URLs 82

Creating or updating score sheets 82

Model, fields, extras and excluded 84

Groups and Match Officials 96

Statistics URL 96

JSON Format 96

# Introduction

This is the API reference manual for Shoot’n Score It (http://shootnscoreit.com).

SSI has many completely open and documented APIs and this manual documents all supported APIs.

Whenever a URL is presented in this document it is assuming the prefix of; https://shootnscoreit.com/api/…

All access to the API is done over https.

SSI does not take any responsibility for any application that access and uses these APIs. All APIs are using CRUD (create, read, update and delete) over POST, GET, PUT and DELETE methods. Each API will state which are supported of these and if there are any requirements.

All JSON documents are returned with mime-type ‘application/json’ and as UTF-8. Authentication is done using HTTP basic authentication and requires/enforces https to be used.

Any comments or questions or ideas for additional features or corrections – please get in touch with us at support@shootnscoreit.com

http://shootnscoreit.com/api/

This API is the same for all supported formats; JSON, XML etc. Currently only JSON is supported.

In the document we outline first an introduction to the API and its methods and various headers and authentication.

Next we define the individual APIs for each type of event (IPSC, USPSA, PPC, etc) and more. For each event we specify the allowed URLs and what they will access. When then also specify the actual data model behind the fields and extras so you can see exactly what the expected in and out data shall be. Note that there is a generic part on events, competitors, etc at the end that specify fields and extras that are common for all.

The format for the fields and excluded and extras are expressed in Python and Django model fields. This means you are actually viewing the true data format that will be serialized and deserialized and this will ensure you get it right. For details on Python/Django model format (it is very easy to read) you can get all the details here; https://www.djangoproject.com and then ‘documentation’ and search for model field reference.

Any comments or questions or ideas for additional features or corrections – please get in touch with us at support@shootnscoreit.com

# Requests and Responses

### Json Responses

All results will be returned as mime-type application/json and in UTF-8.

Each JSON response consists of an array of zero or more elements. Each elements a has a ‘pk’ i.e. a primary key of the instance within SSI, the model as in the unique model name within SSI and then fields and extras.

Fields are individual fields and their value from this instance. Depending on your authorization (see reference manual on roles and authorization) there can different fields shown. The same goes for extra fields.

Extras are extra information of this instance that in most cases is deducted and not stored as explicit fields within the instance yet they are provided since they give valuable information about the data. Here is an example of what it might look like;

[
 {
 "pk": 1,
 "model": "model\_name",
 "ct\_pk": 7,
 "extra\_1": "value",
 …
 "extra\_n": “value”
 "field\_1": "value",
 …
 "field\_n": “value”
 }
]

The pk is the primary key for this instance of the model ‘model\_name’. The ct\_pk is the Content Type primary key or the primary key for the model itself.

### Overview of JSON Request and responses

A request shall follow the same JSON format as provided in a response, i.e. an array of instances.

Accept header must be set to JSON.

Field names request and response are case- sensitive.

If a request is to create one or more new instances this shall be done with a POST and only stating model and fields, extras can be provided but there is no guarantee they will be considered. Primary key shall not be provided. If successful the equivalent results for GET on these instances will be returned.

If a request is to update one or more existing instances this shall be done with a PUT and stating model, primary key and fields, extras can be provided but there is no guarantee they will be considered. If successful the equivalent results for GET on these instances will be returned.

A Http request needs to have accept header set to ‘JSON’, all Http responses will have content type set to application/json and be in UTF-8 format.

### Overview of Request-RESPONSE CODES

|  |  |
| --- | --- |
| Client request | Possible Server response |
| GET | 200 OK, with json400 Bad request, if invalid format or data401 Unauthorized, missing permission404 Not Found |
| PUT | 200 OK, with json400 Bad request, if invalid format or data401 Unauthorized, missing permission404 Not Found |
| POST | 200 OK, with json400 Bad request, if invalid format or data401 Unauthorized, missing permission404 Not Found |
| DELETE | 200 Accepted, found and deleted400 Bad request, if invalid format or data401 Unauthorized, missing permission404 Not Found |
| HEADTRACE… | 405 Method Not Allowed |

# Authentication and Authorization

You can access some information without providing authentication and some require authentication to be returned and some actions also require authentication and authorization.

Shoot’n Score It uses a RBAC (Role Based Access Control) mechanism internally. This means that depending on your role in relation to an event (if you are only someone viewing it, if you are a competitor in it, if you a an organizer as admin or staff) you will be allowed to see various levels of information and perform different actions. E.g. only an organizer can enter scores for an event, only you yourself can updated your personal settings and so on.

This means that in many cases you will need to be authenticated towards SSI in order to be able to perform some actions or you can expect to a 401 Unauthenticated or 404 Not Found response.

If you wish to authenticate this is done over HTTPS (SSL/TLS) using Http Basic Authentication, i.e. you provide username (email) and password for the user/shooter over a secure connection. This is easy to do and supported on all browsers and platforms.

For more details on Http Basic Authentication see:

http://en.wikipedia.org/wiki/Basic\_access\_authentication

For details on Authorization and what to expect see reference manual and ‘group’ concept that is administered in SSI.

## Implementation tricks

If you make a request we recommend you set the X\_REQUESTED\_WITH header to XMLHttpRequest, then if not authenticated the API will give a response with WWW-Authenticate header set to ‘Basic/WebApp’, if no or other X\_REQUESTED\_WITH header then the authenticate header will just be set to ‘Basic’. This allows you to do AJAX calls from a browser without the browser login window appearing (e.g. in a HTML5 standalone client).Tips and Tricks

## Getting started using API

It is really useful for you to start testing accessing this REST API using some simple tool that allows you to hand-craft your requests.

To get examples you can edit and test just access the API URL from an ordinary browser.

For Apple/OS X we recommend e.g. REST Tester or RESTed (simple applications that you can use). For Windows there are several similar applications or you can always use some browser add-ons like Poster for Firefox or REST Console for Chrome. Then there is always cURL that allows you to do anything.

For inspiration and examples just google ‘test REST API’ and you will find lots and lots of examples on how to interact with a REST API and tools.

For some parts of the API a valid premium-shooter account is required. If you are developing a client or app – just email us and we will upgrade your developer shooter account to premium so you can develop stuff.

## L18N and I13n

Any application using this API is strongly recommended to support Localization and Internationalization. The user settings can provide information about language and region of shooter if needed.

## User-Agent

Whenever you develop an app that uses SSI, please set user-agent header to something that makes sense – see <http://en.wikipedia.org/wiki/User_agent>

This will allow us to present and get stats on the usage and if there is any issues with some user agent and version.

## Synchronization and performance

Each instance has a created and updated key and value as a ISO8601 date and time + timezone info. This can be used to determine if the instance has been updated or not. The format is %Y-%m-%dT%H:%M:%SZ (where Z is timezone) or e.g. "2012-07-19T10:00:00Z".

Each valid response to API also has a X-Server-Datetime that returns the current date and time on server. In case you fetch many instances, and need to do this call again you can then only query for instance that has been updated after the last server datetime (since you already have fetched all that existed at this time).

There is a ‘…/all/’ URL for all types of events. For a match this will return the match information, all competitors, all stages and all scorecards. For a series it is series information, all competitors and all component matches.

# Intro to data reference

In the following sections starts with defining the individual URLs you can use to access information over the API. We then define the data formats in Python/Django model formats and show the individual choices available and supported.

Fields are actual fields. Depending on your authorization (are you competitor, admin or staff) it might that some fields are excluded for you.

Extras are information that is deducted and not stored as fields, or might be easier to access as deducted information than the separate raw data.

## Query format

In some URL you can request the objects to be returned, to be filtered on some conditions.

https://…../?updated\_gte=2012-07-19T10:00:00Z

Currently the following is supported;

updated\_gt – updated greater then a ISO8601 timezone aware date and time

updated\_gte – updated greater then or equal ISO8601 timezone aware date and time

updated\_lt – updated less then ISO8601 timezone aware date and time

updated\_lte – updated less then or equal ISO8601 timezone aware date and time

URLs that support the query format is marked in document.

# General Features

### Search for Events

/search\_event/ [POST]

Post a JSON body with the following format to search for events. If you do not have a value or wishes to search for ‘any’ then leave out the field and value (not as just an empty string)

{

 "fields": {

 "rule": "ip",

 "name": "BlÃ¶w",

 "starts\_after": "2012-01-19T16:21:05Z",

 "starts\_before": "2012-06-19T16:21:05Z",

 "status": "om",

 "max\_hits": 100,

 "region": "SWE

 }

 "show": {

 "count": true,

 "max\_hits": 25,

 "start\_index": 30,

 "stop\_index": 50,

 }

}

The ‘Fields’ expresses the search criteria;

* rule; searching for a specific match rule (see match choices), if no field any rule searched
* status; only events in this state (see match state choices), if no field any state
* starts\_before and starts\_after; date interval, must be on format YYYY-MM-DDTHH:MM TZ, if left out any date assumed
* name; searches for any event with this as substring in name and case insensitive
* region; searches for matches in this region, must be ISO 3166 3 letter region, see 3166 listing supported.

Results are always returned ascending (default, see reverse in ‘result’ below).

The ‘show’ expresses how results should be expressed with the :

* count; default False. optional, if set to True then the result returned will be number of hits and nothing else. Default if False
* max\_hits: max hits you want to get back (positive integer), default to internal setting (20)
* index\_start: returns all results starting from this index (positive or negative index in list), default 0
* index\_stop: returns all results up to this index (positive or negative index in list), default -1
* reverse: default True, can be set to False and then results will be sorted in ‘start date’ descending order.

All fields in show are optional. Count has precedence, indexes has precedence over max hits.

# API is Alive

/alive/ – always returns 200 OK. Can be used by client as a simple way to verify if they have access to API and can make calls or sync.

# Shooters

## Sign-up or bulk-creating users/shooters

You can create users directly towards the API to this URL and with the JSON body below.

/signup/ [POST] – create a user over API.

The JSON body for creating a user is a a dictionary with key ‘user’ and ‘profile’ for two other dictionaries and the following attributes:

{

 "user": {

 "first\_name": "Jens",

 "last\_name": "Lundstrom",

 "email": "jens@acme.com",

 "pwd1": "my-new-password"

 },

 "profile": {

 "region": "USA",

 "sex": "m",

 "timezone": "UTC"

 }

}

The ordering of the two dictionaries are important, and all key/value pairs must be included. Strongly recommend posting this over https. No authentication is needed. The possible response codes are as follows:
 200 - user created successfully

 405 - you are not doing a POST, this URL only supports POST method.

 409 - user with this email already exist

 500 - some error with the data, or some other internal issue, see the message body for details

 503 - service not available

This makes it easy for you to create a script (see examples available at SSI where there are JSON and SCV files you can edit and use) for bulk-creating users for e.g. all your members of your club or to integrate this with your own web site sign-up.

## General information for myself

/myself/ [GET PUT POST] – returns my information or settings for user and profile. If POST or PUT provide full profile or

/my\_registrations/ [GET] – returns competitors for ongoing events

/my\_events/ [GET] - returns matches and series

/my\_results/ [GET] – returns competitors for completed events

## Change password

/new\_password/ [PUT POST] - changes password for current user to the password provided.

Any check (like entering password twice and matching must be done in client). Any logged in session for this user in any web browser will be cancelled. Note that password is string of at least 5 characters

[ { "password": "XXXXXX" } ]

## Verifying score using PIN or password

All score sheets has a ‘verified\_by’ and ‘verified’ (datetime). Only the competitor can verify his or her own scorecards and an organizer un-verify. See URL for score sheets in each sport.

Verification can be done in two ways; first you can POST to an API verify URL for your own scorecards and as you are the competitor for the scorecard it will be verified, secondly anyone can POST to the same URL with a form with field and value ‘password’ for the competitor associated with this scorecard or ‘PIN’.

This last allows for an RO to be logged in and authenticate and have competitor enter their password or PIN as verification in client, then POST is done and authenticated for RO and as RO is not competitor the API will look for a password or PIN in the form and check that against the competitor and if a match – mark as verified by the competitors him- or her-self. A well-behaving client will not log the competitor password or PIN in any manner.

The difference between PIN and password is that password is that of the shooter (i.e. his/her login to SSI) but PIN is a 4-digit code only associated with the competitor (not with the shooter/user in SSI). The PIN is never set for a competitors in an event, instead the PIN will be set for the competitor the first time the competitor verifies a scorecard. This means that the first verify POST for a competitor with a PIN will both set the PIN to be used within the event for the competitor as well as verify the scorecard. Given the limited usage of a PIN only within an event and that the POST in itself must be authenticated (and logged) provides enough security for it to be used – as this means there will never be any need for a competitor to choose PIN prior to event (and handle cases where they have forgotten it).

In case of a forgotten PIN, the competitor or organizer can simply reset it and it will be set the next time it is used. The competitor is also allowed to set it at any stage (and this will replace previous setting of PIN in the event).

Un-verification for a scorecard is only allowed by under RBAC role ‘manage competitor’ (i.e. you need to be admin for event). PIN reset is allowed by the competitor and the event organizer.

Un-verification will also always happen explicitly if a score is verified and then is updated afterwards.

## Model, fields, extras and excluded

### Match and series

See generic information

### Competitors

See generic information.

### User and Profile

CURRENCIES\_ISO\_4217 = (

 ('EUR', \_(u'Euro (EUR)')),

 ('USD', \_(u'U.S. Dollar (USD)')),

 ('AUD', \_(u'Australian Dollar (AUD)')),

 ('CAD', \_(u'Canadian Dollar (CAD)')),

 ('CHF', \_(u'Swiss Franc (CHF)')),

 ('CZK', \_(u'Czech Koruna (CZK)')),

 ('DKK', \_(u'Danish Krone (DKK)')),

 ('GBP', \_(u'Pound Sterling (GBP)')),

 ('HKD', \_(u'Hong Kong Dollar (HKD)')),

 ('HUF', \_(u'Hungarian Forint (HUF)')),

 ('ILS', \_(u'Israeli New Shekel (ILS)')),

 ('JPY', \_(u'Japanese Yen (JPY)')),

 ('MXN', \_(u'Mexican Peso (MXN)')),

 ('NOK', \_(u'Norwegian Krone (NOK)')),

 ('NZD', \_(u'New Zealand Dollar (NZD)')),

 ('PHP', \_(u'Philippine Peso (PHP)')),

 ('PLN', \_(u'Polish Zloty (PLN)')),

 ('RUB', \_(u'Russian Ruble (RUB)')),

 ('SEK', \_(u'Swedish Krona (SEK)')),

 ('SGD', \_(u'Singapore Dollar (SGD)')),

 ('ZAR', \_(u'South African Rand (ZAR)')),

 ('THB', \_(u'Thai Bhat (THB)')),

 ('TWD', \_(u'New Taiwan Dollar (TWD)')),

)

# languages are on format 'name-in-local-language / translated-to-current-language'

LANGUAGE\_CHOICES = (

 ('en', \_('English')),

 ('sv', \_('Swedish')),

 ('sq', \_('Albanian')),

 ('ar', \_('Arabic')),

 ('eu', \_('Basque')),

 ('bg', \_('Bulgarian')),

 ('bn', \_('Bengali')),

 ('zh\_CN', \_('Chinese (China)')),

 ('cs', \_('Czech')),

 ('km', \_('Cambodian')),

 ('ca', \_('Catalan')),

 ('hr', \_('Croatian')),

 ('da', \_('Danish')),

 ('nl', \_('Dutch')),

 ('et', \_('Estonian')),

 ('fi', \_('Finnish')),

 ('fr', \_('French')),

 ('gl', \_('Galican')),

 ('ka', \_('Georgian')),

 ('de', \_('German')),

 ('el', \_('Greek')),

 ('he', \_('Hebrew')),

 ('hi', \_('Hindi')),

 ('hu', \_('Hungarian')),

 ('is', \_('Icelandic')),

 ('ga', \_('Irish')),

 ('id', \_('Indonesian')),

 ('it', \_('Italian')),

 ('ja', \_('Japanese')),

 ('ko', \_('Korean')),

 ('lv', \_('Latvian')),

 ('lt', \_('Lithuanian ')),

 ('mk', \_('Macedonian')),

 ('mn', \_('Mongolian')),

 ('ml', \_('Malayalam')),

 ('no', \_('Norwegian')),

 ('fa', \_('Persian')),

 ('pt', \_('Portuguese')),

 ('pl', \_('Polish')),

 ('ru', \_('Russian')),

 ('ro', \_('Romanian')),

 ('sr', \_('Serbian')),

 ('es', \_('Spanish')),

 ('sl', \_('Slovenian')),

 ('sk', \_('Slovak')),

 ('th', \_('Thai')),

 ('tr', \_('Turkish')),

 ('uk', \_('Ukranian')),

 ('vi', \_('Vietnamese')),

 ('cy', \_('Welsh')),

 )

REGIONS\_3166A3 = (

 ('ALA', \_(u'Aland Islands')),

 ('AFG', \_(u'Afghanistan')),

 ('ALB', \_(u'Albania')),

 ('DZA', \_(u'Algeria')),

 ('ASM', \_(u'American Samoa')),

 ('AND', \_(u'Andorra')),

 ('AGO', \_(u'Angola')),

 ('AIA', \_(u'Anguilla')),

 ('ATA', \_(u'Antarctica')),

 ('ATG', \_(u'Antigua and Barbuda')),

 ('ARG', \_(u'Argentina')),

 ('ARM', \_(u'Armenia')),

 ('ABW', \_(u'Aruba')),

 ('AUS', \_(u'Australia')),

 ('AUT', \_(u'Austria')),

 ('AZE', \_(u'Azerbaijan')),

 ('BHS', \_(u'Bahamas')),

 ('BHR', \_(u'Bahrain')),

 ('BGD', \_(u'Bangladesh')),

 ('BRB', \_(u'Barbados')),

 ('BLR', \_(u'Belarus')),

 ('BEL', \_(u'Belgium')),

 ('BLZ', \_(u'Belize')),

 ('BEN', \_(u'Benin')),

 ('BMU', \_(u'Bermuda')),

 ('BTN', \_(u'Bhutan')),

 ('BOL', \_(u'Bolivia')),

 ('BIH', \_(u'Bosnia and Herzegowina')),

 ('BWA', \_(u'Botswana')),

 ('BVT', \_(u'Bouvet Island')),

 ('BRA', \_(u'Brazil')),

 ('IOT', \_(u'British Indian Ocean Territory')),

 ('BRN', \_(u'Brunei Darussalam')),

 ('BGR', \_(u'Bulgaria')),

 ('BFA', \_(u'Burkina Faso')),

 ('BDI', \_(u'Burundi')),

 ('KHM', \_(u'Cambodia')),

 ('CMR', \_(u'Cameroon')),

 ('CAN', \_(u'Canada')),

 ('CPV', \_(u'Cape Verde')),

 ('CYM', \_(u'Cayman Islands')),

 ('CAF', \_(u'Central African Republic')),

 ('TCD', \_(u'Chad')),

 ('CHL', \_(u'Chile')),

 ('CHN', \_(u'China')),

 ('CXR', \_(u'Christmas Island')),

 ('CCK', \_(u'Cocos (Keeling) Islands')),

 ('COL', \_(u'Colombia')),

 ('COM', \_(u'Comoros')),

 ('COG', \_(u'Congo')),

 ('COD', \_(u'Democratic Rep of Congo')),

 ('COK', \_(u'Cook Islands')),

 ('CRI', \_(u'Costa Rica')),

 ('CIV', \_(u'Cote D\'Ivoire')),

 ('HRV', \_(u'Croatia')),

 ('CUW', \_(u'Curacao')),

 ('CUB', \_(u'Cuba')),

 ('CYP', \_(u'Cyprus')),

 ('CZE', \_(u'Czech Republic')),

 ('DNK', \_(u'Denmark')),

 ('DJI', \_(u'Djibouti')),

 ('DMA', \_(u'Dominica')),

 ('DOM', \_(u'Dominican Republic')),

 ('ECU', \_(u'Ecuador')),

 ('EGY', \_(u'Egypt')),

 ('SLV', \_(u'El Salvador')),

 ('GNQ', \_(u'Equatorial Guinea')),

 ('ERI', \_(u'Eritrea')),

 ('EST', \_(u'Estonia')),

 ('ETH', \_(u'Ethiopia')),

 ('FLK', \_(u'Falkland Islands (Malvinas)')),

 ('FRO', \_(u'Faroe Islands')),

 ('FJI', \_(u'Fiji')),

 ('FIN', \_(u'Finland')),

 ('FRA', \_(u'France')),

 ('GUF', \_(u'French Guiana')),

 ('PYF', \_(u'French Polynesia')),

 ('ATF', \_(u'French Southern Territories')),

 ('GAB', \_(u'Gabon')),

 ('GMB', \_(u'Gambia')),

 ('GEO', \_(u'Georgia')),

 ('DEU', \_(u'Germany')),

 ('GHA', \_(u'Ghana')),

 ('GIB', \_(u'Gibraltar')),

 ('GRC', \_(u'Greece')),

 ('GRL', \_(u'Greenland')),

 ('GRD', \_(u'Grenada')),

 ('GLP', \_(u'Guadeloupe')),

 ('GUM', \_(u'Guam')),

 ('GTM', \_(u'Guatemala')),

 ('GGY', \_(u'Guernsey')),

 ('GIN', \_(u'Guinea')),

 ('GNB', \_(u'Guinea-Bissau')),

 ('GUY', \_(u'Guyana')),

 ('HTI', \_(u'Haiti')),

 ('HMD', \_(u'Heard and Mc Donald Islands')),

 ('VAT', \_(u'Holy See (Vatican city state)')),

 ('HND', \_(u'Honduras')),

 ('HKG', \_(u'Hong kong')),

 ('HUN', \_(u'Hungary')),

 ('ISL', \_(u'Iceland')),

 ('IND', \_(u'India')),

 ('IDN', \_(u'Indonesia')),

 ('IRN', \_(u'Iran')),

 ('IRQ', \_(u'Iraq')),

 ('IRL', \_(u'Ireland')),

 ('IMN', \_(u'Isle of man')),

 ('ISR', \_(u'Israel')),

 ('ITA', \_(u'Italy')),

 ('JAM', \_(u'Jamaica')),

 ('JPN', \_(u'Japan')),

 ('JEY', \_(u'Jersey')),

 ('JOR', \_(u'Jordan')),

 ('KAZ', \_(u'Kazakhstan')),

 ('KEN', \_(u'Kenya')),

 ('KIR', \_(u'Kiribati')),

 ('KOR', \_(u'Korea, Republic of')),

 ('KWT', \_(u'Kuwait')),

 ('KGZ', \_(u'Kyrgyzstan')),

 ('LVA', \_(u'Latvia')),

 ('LAO', \_(u'Laos')),

 ('LBN', \_(u'Lebanon')),

 ('LSO', \_(u'Lesotho')),

 ('LBR', \_(u'Liberia')),

 ('LBY', \_(u'Libyan Arab Jamahuriya')),

 ('LIE', \_(u'Liechtenstein')),

 ('LTU', \_(u'Lithuania')),

 ('LUX', \_(u'Luxembourg')),

 ('MAC', \_(u'Macao')),

 ('MKD', \_(u'Macedonia')),

 ('MDG', \_(u'Madagascar')),

 ('MWI', \_(u'Malawi')),

 ('MYS', \_(u'Malaysia')),

 ('MDV', \_(u'Maldives')),

 ('MLI', \_(u'Mali')),

 ('MLT', \_(u'Malta')),

 ('MHL', \_(u'Marshall Islands')),

 ('MTQ', \_(u'Martinique')),

 ('MRT', \_(u'Mauritania')),

 ('MUS', \_(u'Mauritius')),

 ('MYT', \_(u'Mayotte')),

 ('MEX', \_(u'Mexico')),

 ('FSM', \_(u'Micronesia')),

 ('MDA', \_(u'Moldova')),

 ('MCO', \_(u'Monaco')),

 ('MNG', \_(u'Mongolia')),

 ('MNE', \_(u'Montenegro')),

 ('MSR', \_(u'Montserrat')),

 ('MAR', \_(u'Morocco')),

 ('MOZ', \_(u'Mozambique')),

 ('MMR', \_(u'Myanmar')),

 ('NAM', \_(u'Namibia')),

 ('NRU', \_(u'Nauru')),

 ('NPL', \_(u'Nepal')),

 ('NLD', \_(u'Netherlands')),

 ('ANT', \_(u'Netherlands Antilles')),

 ('NCL', \_(u'New Caledonia')),

 ('NZL', \_(u'New Zealand')),

 ('NIC', \_(u'Nicaragua')),

 ('NER', \_(u'Niger')),

 ('NGA', \_(u'Nigeria')),

 ('NIU', \_(u'Niue')),

 ('NFK', \_(u'Norfolk Island')),

 ('MNP', \_(u'Northern Mariana Islands')),

 ('NOR', \_(u'Norway')),

 ('OMN', \_(u'Oman')),

 ('PAK', \_(u'Pakistan')),

 ('PLW', \_(u'Palau')),

 ('PAN', \_(u'Panama')),

 ('PNG', \_(u'Papua New Guinea')),

 ('PRY', \_(u'Paraguay')),

 ('PER', \_(u'Peru')),

 ('PHL', \_(u'Philippines')),

 ('PCN', \_(u'Pitcairn')),

 ('POL', \_(u'Poland')),

 ('PRT', \_(u'Portugal')),

 ('PRI', \_(u'Puerto Rico')),

 ('QAT', \_(u'Qatar')),

 ('REU', \_(u'Reunion')),

 ('ROU', \_(u'Romania')),

 ('RUS', \_(u'Russian')),

 ('RWA', \_(u'Rwanda')),

 ('KNA', \_(u'Saint Kitts/Nevis')),

 ('LCA', \_(u'Saint Lucia')),

 ('VCT', \_(u'Saint Vincent and te Grenadines')),

 ('WSM', \_(u'Samoa')),

 ('SMR', \_(u'San Marino')),

 ('STP', \_(u'Sao Tome and Principe')),

 ('SAU', \_(u'Saudi Arabia')),

 ('SEN', \_(u'Senegal')),

 ('SRB', \_(u'Republic of Serbia')),

 ('SYC', \_(u'Seychelles')),

 ('SLE', \_(u'Sierra Leone')),

 ('SGP', \_(u'Singapore')),

 ('SVK', \_(u'Slovakia')),

 ('SVN', \_(u'Slovenia')),

 ('SLB', \_(u'Solomon Islands')),

 ('SOM', \_(u'Somalia')),

 ('ZAF', \_(u'South Africa')),

 ('SGS', \_(u'South Georgia')),

 ('ESP', \_(u'Spain')),

 ('LKA', \_(u'Sri Lanka')),

 ('SHN', \_(u'St. Helena')),

 ('SPM', \_(u'St. Pierre and Miquelon')),

 ('SDN', \_(u'Sudan')),

 ('SUR', \_(u'Suriname')),

 ('SJM', \_(u'Svalbard')),

 ('SWZ', \_(u'Swaziland')),

 ('SWE', \_(u'Sweden')),

 ('CHE', \_(u'Switzerland')),

 ('SYR', \_(u'Syrian')),

 ('TWN', \_(u'Taiwan')),

 ('TJK', \_(u'Tajikistan')),

 ('TZA', \_(u'Tanzania')),

 ('THA', \_(u'Thailand')),

 ('TLS', \_(u'Timor-leste')),

 ('TGO', \_(u'Togo')),

 ('TKL', \_(u'Tokelau')),

 ('TON', \_(u'Tonga')),

 ('TTO', \_(u'Trinidad and Tobago')),

 ('TUN', \_(u'Tunisia')),

 ('TUR', \_(u'Turkey')),

 ('TKM', \_(u'Turkmenistan')),

 ('TCA', \_(u'Turks and Caicos Islands')),

 ('TUV', \_(u'Tuvalu')),

 ('UGA', \_(u'Uganda')),

 ('UKR', \_(u'Ukraine')),

 ('ARE', \_(u'United Arab Emirates')),

 ('GBR', \_(u'United Kingdom')),

 ('USA', \_(u'United States')),

 ('URY', \_(u'Uruguay')),

 ('UZB', \_(u'Uzbekistan')),

 ('VUT', \_(u'Vanuatu')),

 ('VEN', \_(u'Venezuela')),

 ('VNM', \_(u'Vietnam')),

 ('VGB', \_(u'Virgin Islands (British)')),

 ('VIR', \_(u'Virgin Islands (U.S.)')),

 ('WLF', \_(u'Wallis and Futuna Islands')),

 ('ESH', \_(u'Western Sahara')),

 ('YEM', \_(u'Yemen')),

 ('ZMB', \_(u'Zambia')),

 ('ZWE', \_(u'Zimbabwe'))

 )

# mapping of 3166A3 CC to IPSC regions codes as of July 2011

REGIONS\_3166A3\_TO\_IPSC = { 'ARG': 'ARG',

 ABW': 'ARU',

 AUS': 'AUS',

 AUT': 'AUT',

 BRB': 'BAR',

 BLR': 'BLR',

 BEL': 'BEL',

 BOL': 'BOL',

 BIH': 'BIH',

 BRA': 'BRA',

 BGR': 'BUL',

 CAN': 'CAN',

 JEY': 'CNI',

 CHL': 'CHI',

 CHN': 'CHN',

 COL': 'COL',

 CRI': 'CRC',

 HRV': 'CRO',

 CUW': 'CUW',

 CZE': 'CZE',

 DNK': 'DEN',

 ECU': 'ECU',

 EGY': 'EGY',

 SLV': 'ESA',

 EST': 'EST',

 FIN': 'FIN',

 FRA': 'FRA',

 DEU': 'GER',

 GRC': 'GRE',

 GUM': 'GUM',

 GTM': 'GUA',

 GUY': 'GUY',

 HND': 'HON',

 HKG': 'HKG',

 HUN': 'HUN',

 IDN': 'INA',

 IRL': 'IRL',

 IMN': 'IOM',

 ISR': 'ISR',

 ITA': 'ITA',

 JAM': 'JAM',

 JPN': 'JPN',

 KAZ': 'KAZ',

 KWT': 'KUW',

 LAO': 'LAO',

 LVA': 'LAT',

 LIE': 'LIE',

 LTU': 'LTU',

 MAC': 'MAC',

 MYS': 'MAS',

 MLT': 'MLT',

 MDA': 'MDA',

 MCO': 'MON',

 MNE': 'MTO',

 NAM': 'NAM',

 NLD': 'NED',

 NZL': 'NZL',

 NIC': 'NCA',

 NOR': 'NOR',

 PNG': 'PNG',

 PRY': 'PAR',

 PER': 'PER',

 PHL': 'PHI',

 POL': 'POL',

 PRT': 'POR',

 PRI': 'PUR',

 ROU': 'ROM',

 RUS': 'RUS',

 LCA': 'LCA',

 SRB': 'SER',

 SGP': 'SIN',

 SVK': 'SVK',

 SVN': 'SLO',

 ZAF': 'RSA',

 ESP': 'ESP',

 LKA': 'SRI',

 SUR': 'SUR',

 SWE': 'SWE',

 CHE': 'SUI',

 TWN': 'TPE',

 THA': 'THA',

 UKR': 'UKR',

 GBR': 'GBR',

 USA': 'USA',

 URY': 'URU',

 VEN': 'VEN',

 ZWE': 'ZIM',

 }

### User

API\_USER\_FIELDS = ( 'first\_name', 'last\_name', 'email')

API\_USER\_EXTRAS = ()

API\_USER\_EXCLUDES = ()

first\_name = models.CharField(\_('first name'), max\_length=30, blank=True)

last\_name = models.CharField(\_('last name'), max\_length=30, blank=True)

email = models.EmailField(\_('e-mail address'), blank=True)

### Profile

# ---------------------------------------------------------------------------

# User Profile

# ---------------------------------------------------------------------------

SEX\_CHOICES = (('m', \_('Male')), ('f', \_('Female')))

VISIBILITY\_CHOICES = ( ('pub', \_('Public, your information is visible and searchable')),

 ('res', \_('Restricted, you are searchable but full details not visible')),

 ('pri', \_('Private, your will not be searchable and no details shown.')))

API\_PROFILE\_FIELDS = ( 'user',

 sex',

 language',

 timezone',

 region',

 club'

 license',

 currency',

 visibility',

 state',

 ics\_alias',

 uspsa\_num',

 idpa\_num',

 waid',
 'sass\_num',

 sass\_alias', )

API\_PROFILE\_EXTRAS = ('get\_phone\_display',)

API\_PROFILE\_EXCLUDES = ('get\_phone\_display', 'license')

user = models.ForeignKey(User, verbose\_name=\_(u'User'), unique=True)

sex = models.CharField(\_(u'Sex'), max\_length=1, choices=SEX\_CHOICES, default='m')

# language is both language and locale

language = models.CharField(\_(u'Language'), max\_length=5, choices=LANGUAGE\_CHOICES, default='en')

timezone = timezones.fields.TimeZoneField(\_('Timezone'), default='Europe/Stockholm')

region = models.CharField(\_(u'Region'), max\_length=3, choices=REGIONS\_3166A3, default='SWE')

club = models.CharField(\_(u'Club'), max\_length=50, blank=True)

license = models.CharField(\_(u'License No'), max\_length=30, blank=True)

currency = models.CharField(\_(u'Currency'), max\_length=3, choices=CURRENCIES\_ISO\_4217, default='EUR')

\_phone = models.CharField(\_(u'Phone'), max\_length=30, blank=True)

visibility = models.CharField(\_(u'Visibility'), max\_length=3, choices=VISIBILITY\_CHOICES, default='pub')

state = models.CharField(\_(u'State'), blank=True, max\_length=3, choices=US\_STATE\_CHOICES)

# country will be region that already exists

 '''

 ICS Alias is a worldwide ID within IPSC

 The ICS alias is not case sensitive (=stored as lower case)

 It must be from 4 to 16 alphanumeric characters (A-Z and 0-9 only, no spaces or special characters).

 '''

ics\_alias = models.CharField(\_(u'ICS alias'), max\_length=16, blank=True)

 '''

 USPSA number issued by USPSA and separate from ICS alias.

 format is e.g. L992349

 \* A - Annual

 \* B - Benefactor

 \* CA - Charter Annual

 \* CAL - Charter Annual Life - CA members who have converted to Life members. This allows CA members to upgrade to life without losing a "super cool" member number

 \* CL - Charter Life

 \* F - Foreign

 \* FL - Foreign Life

 \* FY - Five Year

 \* FYF - Five Year Foreign

 \* HCL - Honorary Chairman Life, issued to Colonel Jeff Cooper

 \* L - Life

 \* RD - Regional Director - USPSA President the get an RD number and get to keep it for the remainder of their life membership.

 \* S - Sponsor - Not sure of the details - there is one member with an "S" number

 \* TY - Three Year

 \* TYF - Three Year Foreign

 '''

 uspsa\_num = models.CharField(\_(u'USPSA number'), max\_length=10, blank=True)

 '''

 IDPDA number

 the membership number in IDPA org. It is a 6 character string, not sure of formatting

 but seems to be A00001 or F000001 where A is American and F is Foreigner.

 '''

 idpa\_num = models.CharField(\_(u'IDPA number'), max\_length=6, blank=True)

 '''

 World Association PPC 1500 ID

 is just a regular number - unknown standard but seems to be only digits and max 5 but no standard exists

 '''

 waid = models.CharField(\_(u'WA1500 Id'), max\_length=8, blank=True)

 '''

 SASS number, the membership number in SASS org. It is a 6 character string.

 '''

sass\_num = models.CharField(\_(u'SASS number'), max\_length=6, blank=True)
sass\_alias = models.CharField(\_(u'SASS alias'), max\_length=50, blank=True)

**Extras**

 @property

 def created(self):

 return self.user.date\_joined

# Armoury

The Armoury is for accessing (CRUD) firearms, sights, handloads, ammunition and training log entries, training exercises and training goals.

This part of the API requires that the shooter have a valid premium account (and if you need this for development purposes, just contact us and we will give you this, no worry).

It is essential that you directly become aware that the logentry model is shared over training log entries, exercises and goals and also that for the training log entry there are several types of entries (shooting, shooting-strings, shooting-accuracy, mental, … and so on) and depending on the klass and type all/some fields should be used or not.

There are also fields for input and output units stating if metric, imperial or mixed units is used and stated explanation for what then the corresponding unit expressed for a field is (e.g. velocity can have different units but most likely a shooter wants to use m/s or fps and nothing else).

You can from the role\_names attribute see if you as current user have right to view\_xxx or manage\_xxx instance. At this time IF you current user is same as shooter then you will always have manage\_xxx rights implicitly. It is very likely we will change this to being an explicit role in the future (still trying to figure out if this is a feature or a bug from our side).

But lets start with the URLs you can use to access things. They are self-explaining…

## URLs

The following are the URLs used to access the parts of armoury:

/my\_armoury/ [GET] gets all activites, firearms, sighs, loads, ammo
/log-activities/ [GET, PUT, POST] for all activities
/log-exercises/ [GET, PUT, POST] for all exercises
/log-goals/ [GET, PUT, POST] for all goals
/logentry/(?P<key>\d+)/ [GET, PUT, POST, DELETE]
/firearms\_sights/ [GET, PUT, POST]
/handloads\_ammunitions/ [GET, PUT, POST]
/firearms/ [GET, PUT, POST]
/sights/ [GET, PUT, POST]
/ammunitions/ [GET, PUT, POST]
/handloads/ [GET, PUT, POST]
/firearm/(?P<key>\d+)/$' [GET, PUT, POST, DELETE]
/sight/(?P<key>\d+)/ [GET, PUT, POST, DELETE]
/handload/(?P<key>\d+)/ [GET, PUT, POST, DELETE]
/ammunition/(?P<key>\d+)/ [GET, PUT, POST, DELETE]

## LogEntry

The LogEntry model is used for Activities, Exercises and Goals.. The later individual sections will show which fields are to be used in each case and which are required and optional when updating instances.

Here are all the fields and there choices available

ENTRY\_KLASS = (
 ('ac', u'Activity'), # logging something that has been performed
 ('ex', u'Exercise'), # exercise or drill that can be reused/shared/...
 ('go', u'Goal'), # goal to achive or complete
)

ENTRY\_STATUS = (
 ('dr', u'draft'), # not yet fully populated with info
 ('ac', u'active'), # ongoing, fully populated with info
 ('cp', u'completed'), # done, fully populated with info
 ('ms', u'missed'), # complete with info but missed, has not happened
 ('cs', u'cancelled'), # for some reason not used / has not or will not happen
)

ENTRY\_TYPE = (
 ('cl', u'Cleaning'), # event for cleaning a weapon
 ('sh', u'Shooting'), # general shooting training event, for generic logging
 ('st', u'Shooting strings'), # shooting and recording strings (record time)
 ('ch', u'Chronograph'), # event for chrono handload recipe and/or ammunition
 ('ac', u'Accuracy'), # accuracy shooting with lots and lots of details
 ('ht', u'Hunting'), # general hunting event
 ('dt', u'Duty/Work'), # shooting in Duty or at Work
 ('mt', u'Mental'), # mental training
 ('pt', u'Physical'), # physical training
 ('cp', u'Competition'), # competition shooting, only completed matches.
 ('hl', u'Handloading'), # handloading ammo - will increase amount of ammo type.
 ('sv', u'Service'), # event for servicing or repairing a weapon
)

TRAINING\_SPORT = (
 ('us', u'USPSA'),
 ('ip', u'IPSC'),
 ('sr', u'SRA'),
 ('id', u'IDPA'),
 ('ir', u'ICORE'),
 ('pp', u'PPC'),
 ('ss', u'SASS'),
 ('sc', u'Steel'),
 ('ns', u'NSSF'),
 ('nd', u'Nordic'),
 ('sk', u'Skeet'),
 ('tr', u'Trap'),
 ('sp', u'Sporting'),
 ('xx', u'General'),
)

ATM\_STANDARD = (
 ('I', 'ICAO'),
 ('A', 'ASM'),
)

#
# The Army Standard Metro (ASM) atmosphere, now used only in ballistics
# defines sea-level conditions as 750.000 mmHg of pressure (29.5275 inHg, 99.9918 kPa),
# 59 F (15 C), and 78% humidity. (Ref: U.S. Army Ballistic Research Laboratory,
# U.S. Army Aberdeen Proving Ground)
#
# For ICAO / ISA see http://fisicaatmo.at.fcen.uba.ar/practicas/ISAweb.pdf
#

SHOOTING\_FOCUS = (
 ('sh', u'Shooting'), # general shooting
 ('tc', u'Tactics'), # duty or tactical trainig
 ('cd', u'Cadence'), # speed between shots
 ('ac', u'Accuracy'), # accuracy
 ('sp', u'Speed'), # overall speed
 ('mv', u'Movement'), # moving during shooting
 ('tr', u'Trigger'), # trigger control/breathing
 ('st', u'Stance'), # body-stance
 ('dy', u'Dry-fire'), # keeping things cheap
 ('ff', u'For fun'), # wasting money :-)
)

MENTAL\_FOCUS = (
 ('rx', u'Relaxation'),
 ('cc', u'Concentration'),
 ('mm', u'Memory'),
 ('st', u'Stress handling'),
 ('tc', u'Tactical'),
)

PHYSICAL\_FOCUS = (
 ('st', u'Strength'),
 ('sp', u'Speed'),
 ('en', u'Endurance'),
 ('fx', u'Flexibility'),
)

API\_LOGENTRY\_FIELDS = API\_PREMIUM\_FIELDS + (
 'input\_units',
 'output\_units',
 'klass',
# 'session',
 'exercise',
 'name',
 'description',
 'firearm',
 'sight',
 'recipe',
 'location',
 'ammunition',
 'event',
 'entry\_type',
 'status',
 'sport',
 'shooting\_focus',
 'mental\_focus',
 'physical\_focus',
 'length',
 'rounds',
 'wind\_direction',
 'humidity',
 'weather',
 'sun\_direction',
 'shooter\_latitude',
 'shooter\_longitude',
 'shooter\_altitude',
 'target\_inclination\_deg',
 'target\_heading\_deg'
 'string\_1',
 'string\_2',
 'string\_3',
 'string\_4',
 'string\_5',
 'string\_6',
 'string\_7',
 'string\_8',
 'string\_9',
 'string\_10',
 'string\_comment\_1',
 'string\_comment\_2',
 'string\_comment\_3',
 'string\_comment\_4',
 'string\_comment\_5',
 'string\_comment\_6',
 'string\_comment\_7',
 'string\_comment\_8',
 'string\_comment\_9',
 'string\_comment\_10',
)

API\_LOGENTRY\_EXTRAS = API\_PREMIUM\_EXTRAS + (
 'temp',
 'wind\_speed',
 'air\_pressure',
 'zeroed\_distance',
 'zeroed\_shift\_elevation',
 'zeroed\_shift\_windage',
 'target\_distance',
 'target\_speed',
 'shot\_group\_cc\_shift\_elevation',
 'shot\_group\_cc\_shift\_windage',
 'shot\_group\_size',
 'starts\_date',
 'starts\_time',
 'ends\_date',
 'ends\_time',
)

API\_LOGENTRY\_EXCLUDES = API\_PREMIUM\_EXCLUDES

klass = models.CharField(\_(u'Sort'), max\_length=2, choices=ENTRY\_KLASS, default='ac', db\_index=True)

 # sessions is a parent relationship set if a training is part of a training session, to parent or main entry
session = models.ForeignKey('self', verbose\_name=\_(u'Session'), related\_name='session\_entries', blank=True, null=True)

 # exercise is a recursive relation referencing some exercise / standard training of some form,
 # used to collect and compare results over time for different standard exercises
 exercise = models.ForeignKey('self', verbose\_name=\_(u'Exercise'), related\_name='exercise\_entries', blank=True, null=True)

 input\_units = models.CharField(\_(u'Units'), max\_length=3, choices=INPUT\_UNITS, default='met')
 output\_units = models.CharField(\_(u'Out Units'), max\_length=8, choices=OUTPUT\_UNITS, default='MOA')

 #
 # general info
 #
 name = models.CharField(\_(u'Reference'), max\_length=40, db\_index=True, blank=True)
 description = models.CharField(\_(u'Description'), max\_length=500, blank=True)

 #
 # weapon, sight, ammo and location
 #
 firearm = models.ForeignKey(Firearm, verbose\_name=\_(u'Firearm'), blank=True, null=True)
 sight = models.ForeignKey(Sight, verbose\_name=\_(u'Sight'), blank=True, null=True)
 ammunition = models.ForeignKey(Ammunition, verbose\_name=\_(u'Ammunition'), blank=True, null=True)
 recipe = models.ForeignKey(HandloadRecipe, verbose\_name=\_(u'Recipe'), blank=True, null=True)
 location = models.ForeignKey(Location, verbose\_name=\_(u'Location'), blank=True, null=True)

 #
 # Type of entry
 #
 entry\_type = models.CharField(\_(u'Type'), max\_length=2, choices=ENTRY\_TYPE, default='sh', db\_index=True)
 status = models.CharField(\_(u'Status'), max\_length=2, choices=ENTRY\_STATUS, default='cp', db\_index=True)
 sport = MultiSelectField(\_(u'Sport'), max\_length=400, choices=TRAINING\_SPORT)
 shooting\_focus = MultiSelectField(\_(u'Shooting Focus'), max\_length=400, choices=SHOOTING\_FOCUS)
 mental\_focus = MultiSelectField(\_(u'Mental Focus'), max\_length=400, choices=MENTAL\_FOCUS)
 physical\_focus = MultiSelectField(\_(u'Physical Focus'), max\_length=400, choices=PHYSICAL\_FOCUS)
 length = models.PositiveIntegerField(\_('Length (min)'), default=60) # in minutes
 rounds = models.PositiveIntegerField(\_('#Rounds'), default=0) # rounds fired

 wind\_direction = models.PositiveIntegerField(\_('Wind direction'), default=0) # clock, 12 is straight ahead, 6 is back
 humidity = models.PositiveIntegerField(\_('Relative Humidity (%)'), default=78) # % relative humidity to this temperature
 weather = models.CharField(\_(u'Weather'), max\_length=6, choices=WEATHER, default='sun')
 sun\_direction = models.PositiveIntegerField(\_('Sun direction'), default=0) # 0= not set, clock 12 is straight ahead, 6 is back
 # atmosphere\_std = models.CharField(\_(u'Atmosphere Standard'), max\_length=3, choices=ATM\_STANDARD, default='I') # will be added at later date

 #
 # For accuracy event: distance, preset, target & where did it really hit
 #
 shooter\_latitude = models.DecimalField(\_(u'Shooter latitude'), max\_digits=12, decimal\_places=6, default=0) # degrees
 shooter\_longitude = models.DecimalField(\_(u'Shooter longitude'), max\_digits=12, decimal\_places=6, default=0) # degrees
 shooter\_altitude = models.DecimalField(\_(u'Shoter altitude'), max\_digits=12, decimal\_places=6, default=0) # degrees

 target\_inclination\_deg = models.PositiveIntegerField(\_('Inclination to target (deg)'), default=0) # 0...360 degrees
 target\_heading\_deg = models.PositiveIntegerField(\_('Target heading (deg)'), default=0) # 0...360 degrees

 #
 # For chrono event: chrono
 #
 chrono = models.CharField(\_(u'Chronograph make'), max\_length=300, blank=True)

 #
 # For chrono event: speed of different shots, 25 should be enough...
 # note that we use cm/sec as this is the same as m/s with two decimals or fps with some decimal
 # to convert back and forth without adding / loosing accuracy
 #
 # allows max 30 shots and with two decimals from m/s and speed as 99999 fps ==> 30 x 6 + 30 x 6 ==> and some marginal 500 x 6 = 3000
 #
 chronograph\_cm\_per\_second = models.CommaSeparatedIntegerField(\_(u'Speeds (cm/s)'), max\_length=3500, default=DEFAULT\_CHRONO)

 #
 # Storing strings of shots and allow for individual comments to each string and shot within a string
 #
 # Each time is stored a seconds with two decimals and in ms (so 34.55 sec is stored as 3455 integer)
 # Assuming each string time is max 99.99 sec then each string takes 9999, or 5 characters so we can store
 # a string of 1000 values - this should be enough space
 #
 string\_1 = models.CommaSeparatedIntegerField(\_(u'String 1 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_2 = models.CommaSeparatedIntegerField(\_(u'String 2 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_3 = models.CommaSeparatedIntegerField(\_(u'String 3 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_4 = models.CommaSeparatedIntegerField(\_(u'String 4 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_5 = models.CommaSeparatedIntegerField(\_(u'String 5 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_6 = models.CommaSeparatedIntegerField(\_(u'String 6 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_7 = models.CommaSeparatedIntegerField(\_(u'String 7 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_8 = models.CommaSeparatedIntegerField(\_(u'String 8 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_9 = models.CommaSeparatedIntegerField(\_(u'String 9 (ms)'), max\_length=5000, default=DEFAULT\_STRING)
 string\_10 = models.CommaSeparatedIntegerField(\_(u'String 10 (ms)'), max\_length=5000, default=DEFAULT\_STRING)

 string\_comment\_1 = models.CharField(\_(u'Comment 1'), max\_length=500, blank=True)
 string\_comment\_2 = models.CharField(\_(u'Comment 2'), max\_length=500, blank=True)
 string\_comment\_3 = models.CharField(\_(u'Comment 3'), max\_length=500, blank=True)
 string\_comment\_4 = models.CharField(\_(u'Comment 4'), max\_length=500, blank=True)
 string\_comment\_5 = models.CharField(\_(u'Comment 5'), max\_length=500, blank=True)
 string\_comment\_6 = models.CharField(\_(u'Comment 6'), max\_length=500, blank=True)
 string\_comment\_7 = models.CharField(\_(u'Comment 7'), max\_length=500, blank=True)
 string\_comment\_8 = models.CharField(\_(u'Comment 8'), max\_length=500, blank=True)
 string\_comment\_9 = models.CharField(\_(u'Comment 9'), max\_length=500, blank=True)
 string\_comment\_10 = models.CharField(\_(u'Comment 10'), max\_length=500, blank=True)

Activity

The following fields are to be used in the different entry types for activities

The are expressed as follows:

Field-name metric unit, imperial unit, mixed unit Mandatory or Optional

First comes the name, then the expected / used units for this field. You are free to change this or express this over metric, imperial and mixed and then use the input-field to express this.

Last if this is mandatory as must have a value for creating an instance or optional. Note that

All fields listed must be present and if not mandatory can be set to ‘’ (as in empty).

For default values the see previous model definitions and selection for choices and more details.

### Fields - Shooting in GEneral

entry\_type ‘sh’

description [ ] Optional
firearm pk to firearm Optional
recipe pk to recipe Optional
ammunition pk to ammunition Optional
rounds [ ] Optional
shooting\_focus [ ] Optional
starts\_date [ ] Mandatory
starts\_time [ ] Mandatory
length [ ] Optional
comment [ ] Optional
visibility [ ] Mandatory

Note: timezone assumed same as for shooter, see timezone set for create/updated dates.

Fields - Shooting String

To be added within short.

Fields - Shooting Accuracy

klass ‘ac’ Mandatory

entry\_type Mandatory

description Optional
firearm pk to firearm Optional

sight pk to sight Optional
recipe pk to recipe Optional
ammunition pk to ammunition Optional
target\_distance m/yards/m Optional
rounds integer Optional
shot\_group\_size mm/inch/mm (two decimals) Optional
shot\_group\_cc\_shift\_elevation mm/inch/mm (two decimals) Optional
shot\_group\_cc\_shift\_windage mm/inch/mm (two decimals) Optional
zeroed\_distance m/yards/m (integer) Optional
zeroed\_shift\_elevation mm/inch/mm (two decimals) Optional
zeroed\_shift\_windage mm/inch/mm (two decimals) Optional
temp c/f/c (integer) Optional
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
comment ‘’ Optional
input\_units Mandatory
visibility Mandatory

FIELDs - Shooting Chronograph

klass ‘ac’ Mandatory

entry\_type Mandatory

description
firearm pk to firearm Optional
recipe pk to recipe Optional
ammunition pk to ammunition Optional
chrono Optional
s0…s9 m/s, fps, fps Optional
shot\_group\_size mm, inch, mm (two decimals)Optional
target\_distance m, yards, m (integer) Optional
temp c, f, c (integer) Optional
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
comment Optional
input\_units Mandatory
visibility Mandatory

Fields - Cleaning

klass ‘ac’ Mandatory

entry\_type Mandatory

description Optional
firearm pk to firearm Optional
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
comment Optional
visibility Mandatory

### Fields - Hunting

klass ‘ac’ Mandatory

entry\_type Mandatory

description Optional
firearm pk to firearm Optional
ammunition pk to firearm Optional
rounds Optional
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
comment Optional
visibility Mandatory

**Fields – Competition**

Not yet supported over API

klass ‘ac’ Mandatory

entry\_type Mandatory

### Fields - Mental training

klass ‘ac’ Mandatory

entry\_type Mandatory

description Optional
mental\_focus Mandatory
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
length minutes (integer) Optional
comment Optional
visibility Mandatory

### Fields - Physical training

klass ‘ac’ Mandatory

entry\_type Mandatory

description Optional
physical\_focus Mandatory
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
length minutes (integer) Optional
comment Optional
visibility Mandatory

Fields - Handloading

klass ‘ac’ Mandatory

entry\_type Mandatory

name
recipe
ammunition
rounds
description
comment
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
visibility

Fields - Service of firearm and/or sight

klass ‘ac’ Mandatory

entry\_type Mandatory

description
comment
firearm
sight
starts\_date yyyy-mm-dd Mandatory
starts\_time hh:mm (24h) Mandatory
visibility

## Exercise

To be added.

## Goals

To be added.

## Firearm

### Model, fields, extras and excluded

INPUT\_UNITS = (
 ('met', u'Metric'),
 ('imp', 'Imperial'),
 ('mix', 'Mixed'),
)

OUTPUT\_UNITS = (
 ('MILS', 'MILS'),
 ('MOA', 'MOA'),
 ('Inches', 'Inches'),
)

TWIST = (
 ('L', 'Left'),
 ('R', 'Right'),
)

RIMFIRE\_CALIBERS = (
 ('ri2a', '17 Hornady Mach 2 (.17HM2)'),
 ('ri3a', '17 Hornady Magnum Rimfire (.17HMR)'),
 ('ri4a', '17 Winchester Super Magnum (.17WSM)'),
 ('ri5a', '22 Long Rifle (.22 LR)'),
 ('ri6a', '22 Short'),
 ('ri7a', '22 Winchester Magnum Rimfire (.22 WMR)'),
 ('ri8a', '22 Winchester Rimfire (.22 WRF)'),
)

RIFLE\_CALIBERS = (
 ('rf1a', '17 Hornet'),
 ('rf2a', '17 Remington'),
 ('rf3a', '22 Accelerator'),
 ('rf4a', '22 Hornet'),
 ('rf5a', '22 Cheetah'),
 ('rf5b', '22-250 Remington'),
 ('rf6a', '204 Ruger'),
 ('rf7a', '220 Russian'),
 ('rf7a', '220 Swift (5.56×56mmSR)'),
 ('rf8a', '222 Remington'),
 ('rf9a', '223 Remington'),
 ('rf10a', '223 Wylde'),
 ('rf11a', '5.56x45mm NATO'),
 ('rf12a', '5.45x39'),
 ('rf13a', '224 Weatherby Magnum'),
 ('rf14a', '22-250 Remington'),
 ('rf15a', '220 Swift'),
 ('rf15a', '225 Winchester'),
 ('rf16a', '5.6x52R (22 Savage HP)'),
 ('rf17a', '6mm PPC'),
 ('rf18a', '6mm Norma BR'),
 ('rf19a', '6XC'),
 ('rf20a', '243 Winchester'),
 ('rf21a', '240 Weatherby Magnum'),
 ('rf22a', '250 Savage'),
 ('rf23a', '257 Roberts'),
 ('rf24a', '25-06 Remington'),
 ('rf25a', '257 Weatherby Magnum'),
 ('rf26a', '6.5 Japanese Arisaka'),
 ('rf27a', '6.5 Carcano'),
 ('rf28a', '6.5x54 M-S'),
 ('rf29a', '260 Remington'),
 ('rf30a', '6.5x55 Swedish Mauser'),
 ('rf31a', '6.5-284 Norma'),
 ('rf32a', '6.5 Grendel'),
 ('rf33a', '6.5 Creedmoor'),
 ('rf34a', '264 Winchester Magnum'),
 ('rf35a', '270 Winchester'),
 ('rf36a', '270 WSM'),
 ('rf27a', '270 Weatherby Magnum'),
 ('rf38a', '6.8 SPC (6.8x43mm)'),
 ('rf39a', '7x57R Mauser'),
 ('rf40a', '7x57 Mauser'),
 ('rf41a', '7mm-08 Remington'),
 ('rf42a', '7x65R'),
 ('rf43a', '284 Winchester'),
 ('rf44a', '280 Remington'),
 ('rf45a', '7x64'),
 ('rf46a', '7mm Remington Magnum'),
 ('rf47a', '7mm Weatherby Magnum'),
 ('rf48a', '7mm Shooting Times Westerner'),
 ('rf49a', '7mm Remington Ultra Magnum'),
 ('rf50a', '7x33mm Sako'),
 ('rf51a', '30-30 Winchester'),
 ('rf52a', '7,5x55 Swiss'),
 ('rf53a', '7.62x39mm'),
 ('rf54a', '308 Winchester'),
 ('rf54b', '7.62x51mm NATO'),
 ('rf54c', '308 Norma Magnum (7.62x65mmBR)r'),
 ('rf55a', '7.62x54R - Russian'),
 ('rf56a', '30-06 Springfield'),
 ('rf57a', '300 AAC Blackout (7.62x35mm)'),
 ('rf58a', '300 Holland & Holland Magnum'),
 ('rf59a', '300 Winchester Short Magnum'),
 ('rf60a', '308 Norma Magnum'),
 ('rf61a', '300 Winchester Magnum'),
 ('rf62a', '300 Weatherby Magnum'),
 ('rf63a', '300 Remington Ultra Magnum'),
 ('rf64a', '30-378 Weatherby Magnum'),
 ('rf65a', '7.7 Japanese'),
 ('rf66a', '303 British'),
 ('rf67a', '7,65 Argentine'),
 ('rf68a', '8x57 JRS'),
 ('rf69a', '8x57 JS'),
 ('rf70a', '338-06 A-square'),
 ('rf71a', '338 Winchester Magnum'),
 ('rf72a', '338 Norma Magnum'),
 ('rf73a', '338 Federal'),
 ('rf74a', '340 Weatherby Magnum'),
 ('rf75a', '338 Lapua Magnum'),
 ('rf76a', '338 Remington Ultra Magnum'),
 ('rf77a', '338-378 Weatherby Magnum'),
 ('rf78a', '358 Norma Magnum'),
 ('rf79a', '9.3x57'),
 ('rf80a', '9.3x62'),
 ('rf81a', '9.3x74R'),
 ('rf82a', '375 Holland & Holland Magnum'),
 ('rf83a', '375 Weatherby Magnum'),
 ('rf84a', '375 Remington Ultra Magnum'),
 ('rf85a', '378 Weatherby Magnum'),
 ('rf86a', '375 Ruger'),
 ('rf87a', '375 Weatherby Magnum'),
 ('rf88a', '375 Whelen (.375-06)'),
 ('rf89a', '416 Remington Magnum'),
 ('rf90a', '416 Rigby'),
 ('rf91a', '416 Weatherby Magnum'),
 ('rf92a', '404 Jeffery'),
 ('rf93a', '45-70 Government'),
 ('rf94a', '458 Winchester Magnum'),
 ('rf95a', '458 Lott'),
 ('rf96a', '458 Sabi'),
 ('rf97a', '458 SOCOM'),
 ('rf98a', '460 Weatherby Magnum'),
 ('rf99a', '470 Nitro Express'),
 ('rf100a', '50 BMG (12.7x99mm NATO)'),
 ('rf101a', '505 Magnum Gibbs'),
)

SHOTGUN\_CALIBERS = (
 ('sg10a', '10 Gauge'),
 ('sg12a', '12 Gauge'),
 ('sg16a', '16 Gauge'),
 ('sg20a', '20 Gauge'),
 ('sg28a', '28 Gauge'),
 ('sg410a', '410 Caliber'),
)

HANDGUN\_CALIBERS = (
 ('hg1a', '25 ACP'),
 ('hg2a', '4,6x30'),
 ('hg3a', '5,6x28'),
 ('hg4a', '32 ACP'),
 ('hg5a', '32 S&W'),
 ('hg6a', '32 S&W Long'),
 ('hg7a', '357 Mag'),
 ('hg8a', '357 SIG'),
 ('hg9a', '38 Casull'),
 ('hg10a', '38 (revolver)'),
 ('hg11a', '380 Auto (9mm Kurtz)'),
 ('hg12a', '38 Super'),
 #('hg13a', '38 Super P+'),
 ('hg14a', '38 Special'),
 ('hg15a', '9x18 (Makarov)'),
 ('hg16a', '9mm (Luger/Parabellum/9x19)'),
 ('hg17a', '9x21mm'),
 ('hg18a', '9x25mm Dillon'),
 ('hg19a', '9x23mm (Largo)'),
 ('hg19b', '9x23mm (Winchester)'),
 ('hg20a', '40 S&W'),
 ('hg21a', '10mm Auto'),
 ('hg22a', '10mm Super'),
 ('hg23a', '400 Corbon'),
 ('hg24a', '41 Mag'),
 ('hg25a', '44 Special'),
 ('hg26a', '44 Magnum'),
 ('hg27a', '45 Auto/ACP'),
 ('hg28a', '45 Colt (Revolver)'),
 ('hg29a', '45 Glock (GAP)'),
 ('hg30a', '454 Casull'),
 ('hg31a', '50 Action Express'),
 ('hg32a', '50 Remington'),
 ('hg33a', '500 S&W Special'),
 ('hg34a', '500 S&W Magnum'),
 ('hg35a', '500 Linebaugh'),
)

AIR\_CALIBERS = (
 ('ai2a', '.177 (4.5mm)'),
 ('ai3a', '.22 (5.5mm & 5.6mm)'),
 ('ai4a', '.20 (5mm)'),
 ('ai5a', '.25 (6.35mm)'),
)

FIREARMS\_CALIBERS = HANDGUN\_CALIBERS + RIFLE\_CALIBERS + SHOTGUN\_CALIBERS + AIR\_CALIBERS

FIREARMS\_MAKE = (
 ('f0a', 'Unknown'),
 ('f2a', '1863 Zouave'),
 ('f3a', '1911'),
 ('f4a', '1911 Custom'),
 ('f5a', '2011'),
 ('f6a', '2011 Custom'),
 ('f7a', 'AR-15'),
 ('f8a', 'AR-15 Custom'),
 ('f9a', 'A&R Sales'),
 ('f10a', 'A.A'),
 ('f11a', 'A.A.A'),
 ('f12a', 'AAC'),
 ('f13a', 'Abadie'),
 ('f14a', 'Abbey, Gerorge T'),
 ('f15a', 'Abesser & Merkel'),
 ('f16a', 'Accu-Match'),
 ('f17a', 'Accuracy International'),
 ('f18a', 'Accu-Tek'),
 ('f19a', 'Acha'),
 ('f20a', 'Acme Arms'),
 ('f21a', 'Action Arms'),
 ('f22a', 'Adams Arms'),
 ('f23a', 'Adamy Gebruder'),
 ('f24a', 'Adc'),
 ('f25a', 'Adcor Defense'),
 ('f26a', 'Addax Tactical'),
 ('f27a', 'Adirondack Arms Co'),
 ('f28a', 'Advanced Armament'),
 ('f29a', 'Advantage Arms'),
 ('f30a', 'Aero Precision'),
 ('f31a', 'AFC'),
 ('f32a', 'Agner'),
 ('f33a', 'Aguirre'),
 ('f34a', 'Aguirre Y Aranzabal (AyA)'),
 ('f35a', 'Air Arms'),
 ('f36a', 'Air Match'),
 ('f37a', 'Ajax Army'),
 ('f38a', 'AK-47'),
 ('f39a', 'Akdal'),
 ('f40a', 'AKKAR'),
 ('f41a', 'Alberta Tactical Rifle'),
 ('f42a', 'Alexander Arms'),
 ('f43a', 'Allen, Ethan'),
 ('f44a', 'Alpha Arms'),
 ('f45a', 'AMAC'),
 ('f46a', 'American Arms, Inc.'),
 ('f47a', 'American Defense Systems'),
 ('f48a', 'American Derringer'),
 ('f49a', 'American Gun Co'),
 ('f50a', 'American Industries'),
 ('f51a', 'American International'),
 ('f52a', 'American Western Arms Inc'),
 ('f53a', 'Ames Sword Co'),
 ('f54a', 'AMT'),
 ('f55a', 'Anderson Manufacturing, Inc.'),
 ('f56a', 'Anschutz'),
 ('f57a', 'Antonio Zoli'),
 ('f58a', 'Apalozo Hermanos'),
 ('f59a', 'AR-15'),
 ('f60a', 'Arcus'),
 ('f61a', 'Ardel Engineering & Mfg. Inc.'),
 ('f62a', 'Arisaka'),
 ('f63a', 'Arizaga'),
 ('f64a', 'Arizmendi Zulaica'),
 ('f65a', 'Armalite'),
 ('f66a', 'Armes De Chasse'),
 ('f67a', 'Armi San Marco'),
 ('f68a', 'Arminex Ltd'),
 ('f69a', 'Arminius'),
 ('f70a', 'Armscor'),
 ('f71a', 'Army Armament'),
 ('f72a', 'Arnold Arms'),
 ('f73a', 'Arsa'),
 ('f74a', 'Arsenal, Inc'),
 ('f75a', 'A-Square'),
 ('f76a', 'Astra'),
 ('f77a', 'ATCSA'),
 ('f78a', 'ATI '),
 ('f79a', 'Aubrey, AJ'),
 ('f80a', 'Augusta Machine Works'),
 ('f81a', 'Australian Automatic Arms LTD'),
 ('f82a', 'Australian Military Arms'),
 ('f83a', 'Auto Mag'),
 ('f84a', 'Auto Ordnance'),
 ('f85a', 'Auto Pointer'),
 ('f86a', 'Axtell Rifle Co'),
 ('f87a', 'Azpiri'),
 ('f88a', 'Baer'),
 ('f89a', 'Baford Arms Inc'),
 ('f90a', 'Baikal'),
 ('f91a', 'Bailons Gunmakers LTD'),
 ('f92a', 'Baker Gun & Forging Co'),
 ('f93a', 'Ballard'),
 ('f94a', 'Barnes Precision Machine'),
 ('f95a', 'Barrett'),
 ('f96a', 'Barrett F.A'),
 ('f97a', 'Barrett FA MFG Co'),
 ('f98a', 'Bascaran, Martin A'),
 ('f99a', 'Bauer FA Corp'),
 ('f100a', 'Bayard'),
 ('f101a', 'Bayonne, Manufacture de´armes'),
 ('f102a', 'BCM'),
 ('f103a', 'Bedell Custom'),
 ('f104a', 'Beeman Precision Arms Inc'),
 ('f105a', 'Belgian'),
 ('f106a', 'Benelli'),
 ('f107a', 'Beretta'),
 ('f108a', 'Bertuzzi'),
 ('f109a', 'Bighorn'),
 ('f110a', 'Bilharz Hall & Co'),
 ('f111a', 'Blackwater Armory'),
 ('f112a', 'Blaser'),
 ('f113a', 'Bleiker'),
 ('f114a', 'Bond Arms Inc'),
 ('f115a', 'Borrowed'),
 ('f116a', 'Brasileira De Cartuchos'),
 ('f117a', 'Bravo Company'),
 ('f118a', 'Bren'),
 ('f119a', 'Bretton'),
 ('f120a', 'Brno'),
 ('f121a', 'Brolin Arms'),
 ('f122a', 'Brown Precision, Inc'),
 ('f123a', 'Browning'),
 ('f124a', 'Browning FN'),
 ('f125a', 'Bruchet'),
 ('f126a', 'Bruff, RP'),
 ('f127a', 'Brugger & Thomet'),
 ('f128a', 'BSA Guns Ltd'),
 ('f129a', 'Buco'),
 ('f130a', 'Budischowsky'),
 ('f131a', 'Bul Transmark Ltd'),
 ('f132a', 'Bullard Repeating Arms Co'),
 ('f133a', 'Bulldog Single Shot Pistol'),
 ('f134a', 'Burgess Gun Co'),
 ('f135a', 'Burgsmuller, K'),
 ('f136a', 'Burnside Rifle Co'),
 ('f137a', 'Bushmaster'),
 ('f138a', 'Butterfield, Jesse'),
 ('f139a', 'C. Sharps'),
 ('f140a', 'C.G. Haenel'),
 ('f141a', 'Cabanas, Industias SA'),
 ('f142a', 'Camp Creek Gunworks'),
 ('f143a', 'Campo Giro'),
 ('f144a', 'Canik'),
 ('f145a', 'Carbon Arms'),
 ('f146a', 'Card, SW'),
 ('f147a', 'Carl-Gustav (CG)'),
 ('f148a', 'Carlton, M'),
 ('f149a', 'Case Willard & Co'),
 ('f150a', 'Caspian Arms, Ltd'),
 ('f151a', 'Castarteilli,Carlo'),
 ('f152a', 'Casull Arms Inc'),
 ('f153a', 'Cavalry Arms'),
 ('f154a', 'CBC'),
 ('f155a', 'CZ (Ceskoslovenska Zbrojovka)'),
 ('f156a', 'Chapuis Armes'),
 ('f157a', 'Charles Daly'),
 ('f158a', 'Charmagne Delvigne'),
 ('f159a', 'Charter Arms'),
 ('f160a', 'Chassepot'),
 ('f161a', 'Cheytac'),
 ('f162a', 'Chiappa Firearms'),
 ('f163a', 'Chinese'),
 ('f164a', 'Christensen Arms'),
 ('f165a', 'Churchhill'),
 ('f166a', 'CIL'),
 ('f167a', 'Cimarron'),
 ('f168a', 'Classic Army'),
 ('f169a', 'Classic Doubles'),
 ('f170a', 'Clement, Chas.'),
 ('f171a', 'Clerke Products'),
 ('f172a', 'Club'),
 ('f173a', 'CMMG'),
 ('f174a', 'Cobra'),
 ('f175a', 'Colt'),
 ('f176a', 'Connecticut Valley Arms Co'),
 ('f177a', 'Constable, R'),
 ('f178a', 'Contento / Ventur'),
 ('f179a', 'Continental'),
 ('f180a', 'Coonan Arms'),
 ('f181a', 'CORE Rifle Systems'),
 ('f182a', 'CPA Rifles'),
 ('f183a', 'Crickett'),
 ('f184a', 'Crosman'),
 ('f185a', 'Custom 2011'),
 ('f186a', 'CVA'),
 ('f187a', 'CzechPoint USA'),
 ('f188a', 'Daewoo'),
 ('f189a', 'Daisy'),
 ('f190a', 'Dakin'),
 ('f191a', 'Dakota'),
 ('f192a', 'Dan Arms of America'),
 ('f193a', 'Dan Wesson'),
 ('f194a', 'Dance & Brothers Confederate Revolvers'),
 ('f195a', 'Daniel Defense '),
 ('f196a', 'Dardick Corp'),
 ('f197a', 'Darne, S.A.'),
 ('f198a', 'Daudeteau'),
 ('f199a', 'Davenport Firearms Co'),
 ('f200a', 'Davidson F.A.'),
 ('f201a', 'Davis & Bozeman'),
 ('f202a', 'Davis Industries'),
 ('f203a', 'Davis-Warner Arms Corporation'),
 ('f204a', 'Daw, G.H.'),
 ('f205a', 'Dawson Precision'),
 ('f206a', 'Daystate'),
 ('f207a', 'DBoys'),
 ('f208a', 'Deane-Harding'),
 ('f209a', 'Defense Technology'),
 ('f210a', 'Delaware Machinery'),
 ('f211a', 'Del-ton'),
 ('f212a', 'Demirett, J'),
 ('f213a', 'Demro'),
 ('f214a', 'Deringer Revolver and Pistol Co'),
 ('f215a', 'Deringer,Henry Rifles and Pistols'),
 ('f216a', 'Destroyer Carbine'),
 ('f217a', 'Detonics'),
 ('f218a', 'Deutsche Werke'),
 ('f219a', 'Diamond'),
 ('f220a', 'Diamondback'),
 ('f221a', 'Dickinson, E.L.'),
 ('f222a', 'Dickson'),
 ('f223a', 'Diemaco'),
 ('f224a', 'Dillon'),
 ('f225a', 'Discoll, JB'),
 ('f226a', 'Dominion Arms'),
 ('f227a', 'Domino'),
 ('f228a', 'Dormus'),
 ('f229a', 'Dornheim, G.C.'),
 ('f230a', 'Doublestar'),
 ('f231a', 'Doublestar Corp'),
 ('f232a', 'Downsizer Corporation'),
 ('f233a', 'DPMS'),
 ('f234a', 'Dragunov'),
 ('f235a', 'Dreadnought Industries'),
 ('f236a', 'Drulov'),
 ('f237a', 'DSA, Inc'),
 ('f238a', 'Dumoulin'),
 ('f239a', 'Dusek, F'),
 ('f240a', 'E. Krause im Heilbron'),
 ('f241a', 'E.M.F. Co Inc'),
 ('f242a', 'EAA'),
 ('f243a', 'EAA/Baikal'),
 ('f244a', 'Eagle Arms'),
 ('f245a', 'Echave & Arizmendi'),
 ('f246a', 'Echeverria'),
 ('f247a', 'Echeverria, Star Bonifacio SA'),
 ('f248a', 'Echo1'),
 ('f249a', 'Eclipse'),
 ('f250a', 'Ed Brown'),
 ('f251a', 'Eddystone'),
 ('f252a', 'Edgar Brothers'),
 ('f253a', 'EDM Arms'),
 ('f254a', 'EIG'),
 ('f255a', 'Elgin Cutlass'),
 ('f256a', 'Eliseo'),
 ('f257a', 'Elite Arms'),
 ('f258a', 'Enders, Carl'),
 ('f259a', 'Enfield'),
 ('f260a', 'Enterprise Arms'),
 ('f261a', 'Erichson, G'),
 ('f262a', 'ERMA'),
 ('f263a', 'Errasti, A'),
 ('f264a', 'Escort'),
 ('f265a', 'ESG Gunworks (Custom Build)'),
 ('f266a', 'Espirin, Hermanos'),
 ('f267a', 'Essential Arms'),
 ('f268a', 'Essex Arme'),
 ('f269a', 'Essex Arms'),
 ('f270a', 'Euroarms of America'),
 ('f271a', 'Evans Repeating Rifle Co'),
 ('f272a', 'Evans, JE'),
 ('f273a', 'Excam'),
 ('f274a', 'Excel Industries'),
 ('f275a', 'Excel Arms'),
 ('f276a', 'Excel of America'),
 ('f277a', 'F.A.S.'),
 ('f278a', 'Fabarm'),
 ('f279a', 'Fabbri, Ivo'),
 ('f280a', 'FN (Fabriques Nationale)'),
 ('f281a', 'Falcon Firearms'),
 ('f282a', 'Famars, A&S'),
 ('f283a', 'Fausti'),
 ('f284a', 'Feather Industries Inc'),
 ('f285a', 'Federal Engineering Corp'),
 ('f286a', 'Federal Ordnance Inc'),
 ('f287a', 'FEG'),
 ('f288a', 'Feinwerkbau'),
 ('f289a', 'Femaru'),
 ('f290a', 'Ferlib'),
 ('f291a', 'Filipietta'),
 ('f292a', 'Finnish Lion'),
 ('f293a', 'Firebird'),
 ('f294a', 'Firestar'),
 ('f295a', 'Firestorm'),
 ('f296a', 'Fletcher Bidwell, LLC'),
 ('f297a', 'FM'),
 ('f298a', 'FMK'),
 ('f299a', 'FNH'),
 ('f300a', 'Foehl & Weeks'),
 ('f301a', 'Fogarty'),
 ('f302a', 'Forehand & Wadsworth'),
 ('f303a', 'Fox'),
 ('f304a', 'Franchi'),
 ('f305a', 'Francotte, A.'),
 ('f306a', 'Freedom Arms'),
 ('f307a', 'Frigon'),
 ('f308a', 'Fusion Firearms'),
 ('f309a', 'Fx Airguns'),
 ('f310a', 'Gabilondo Y Urresti'),
 ('f311a', 'Galand, C.F.'),
 ('f312a', 'Galef'),
 ('f313a', 'Galena Industries Inc'),
 ('f314a', 'Galesi, Industia Armi'),
 ('f315a', 'Gamba, Renato'),
 ('f316a', 'Gamo'),
 ('f317a', 'Garate, Anitua/G.A.C.'),
 ('f318a', 'Garate, Hermanos'),
 ('f319a', 'Garbi'),
 ('f320a', 'Garcia'),
 ('f321a', 'Garret, J & F Co'),
 ('f322a', 'Gasser, Leopold'),
 ('f323a', 'Gaucha-Iga'),
 ('f324a', 'Gaucher'),
 ('f325a', 'Gazanga, Isidro'),
 ('f326a', 'Generic .22lr'),
 ('f327a', 'Geoffrey Hayes'),
 ('f328a', 'Gering, H.M. & Co'),
 ('f329a', 'German Sport Guns'),
 ('f330a', 'Gevarm'),
 ('f331a', 'GIB'),
 ('f332a', 'Gibbs Rifle Company'),
 ('f333a', 'Glock'),
 ('f334a', 'Goncz Co'),
 ('f335a', 'GP'),
 ('f336a', 'Grand Power'),
 ('f337a', 'Grand Precision, Gabrique D;armes De'),
 ('f338a', 'Greene'),
 ('f339a', 'Grendel'),
 ('f340a', 'Griffin'),
 ('f341a', 'Grozer'),
 ('f342a', 'GSG'),
 ('f343a', 'Gunworks Ltd'),
 ('f344a', 'Gustloff-Werke'),
 ('f345a', 'Gwyn & Campbell'),
 ('f346a', 'H&R Arms Co.'),
 ('f347a', 'H&K (Heckler & Koch)'),
 ('f348a', 'H&R Arms'),
 ('f349a', 'H.P.S'),
 ('f350a', 'Haenel, C.G.'),
 ('f351a', 'Hafdasa'),
 ('f352a', 'Hahn, William'),
 ('f353a', 'Hale & Tuller'),
 ('f354a', 'Hale, HJ'),
 ('f355a', 'Hall, Alexander'),
 ('f356a', 'Hall-North'),
 ('f357a', 'Hamilton Rifle Company'),
 ('f358a', 'Hammerli'),
 ('f359a', 'Hammerli, SA'),
 ('f360a', 'Hammerli-Walther'),
 ('f361a', 'Hammond Bulldog'),
 ('f362a', 'Harrington & Richardson'),
 ('f363a', 'Harris Gunworks'),
 ('f364a', 'Hartford Arms'),
 ('f365a', 'Hatfield Rifle'),
 ('f366a', 'Hatsan'),
 ('f367a', 'Hatzan'),
 ('f368a', 'Haulikot'),
 ('f369a', 'Haviland & Gunn'),
 ('f370a', 'Hawa'),
 ('f371a', 'Hawes'),
 ('f372a', 'Hawes & Waggoner'),
 ('f373a', 'Hawk'),
 ('f374a', 'Hawkins'),
 ('f375a', 'Hawthorne'),
 ('f376a', 'HDH, SA'),
 ('f377a', 'Heavy Express Inc'),
 ('f378a', 'Heinzelmann, C.E.'),
 ('f379a', 'Helfricht'),
 ('f380a', 'Henrion & Dassy'),
 ('f381a', 'Henry'),
 ('f382a', 'Henry Repeating Arms'),
 ('f383a', 'Heritage Manufacturing Inc'),
 ('f384a', 'Herold'),
 ('f385a', 'Herter´s'),
 ('f386a', 'Hesse Arms'),
 ('f387a', 'Heym, F.W.'),
 ('f388a', 'High Point'),
 ('f389a', 'Hill, W.J.'),
 ('f390a', 'Hilliard, D.H.'),
 ('f391a', 'HJS Industries, Inc'),
 ('f392a', 'Hodge Defense'),
 ('f393a', 'Holland and Holland'),
 ('f394a', 'Holloway Arms Co'),
 ('f395a', 'Holmes Firearms'),
 ('f396a', 'Hopkins & Allen'),
 ('f397a', 'Howa'),
 ('f398a', 'H-S Precision Inc'),
 ('f399a', 'Huglu'),
 ('f400a', 'Huldra Mark 4'),
 ('f401a', 'Hungary'),
 ('f402a', 'Husqvarna'),
 ('f403a', 'Hybrid'),
 ('f404a', 'Hyde & Shattuck'),
 ('f405a', 'Hy-Hunter'),
 ('f406a', 'Hyper'),
 ('f407a', 'IAI-American Legends'),
 ('f408a', 'IGA'),
 ('f409a', 'Ikka'),
 ('f410a', 'IM Metal'),
 ('f411a', 'IMI '),
 ('f412a', 'Indian Arms Corp'),
 ('f413a', 'Inferno'),
 ('f414a', 'Infinity'),
 ('f415a', 'Ingram'),
 ('f416a', 'Inland'),
 ('f417a', 'Inter Ordanance'),
 ('f418a', 'Interarms'),
 ('f419a', 'Interdynamic'),
 ('f420a', 'Intratec'),
 ('f421a', 'IO Inc'),
 ('f422a', 'Irving, W'),
 ('f423a', 'Isreal Arms LTD'),
 ('f424a', 'Ithaca'),
 ('f425a', 'Iver Johnson Arms Inc'),
 ('f426a', 'IWI'),
 ('f427a', 'IXL'),
 ('f428a', 'Izhvesk'),
 ('f429a', 'J C HIggins'),
 ('f430a', 'J. Stevens Arms Company'),
 ('f431a', 'J.R. Carbine'),
 ('f432a', 'Jacquesmart, Jules'),
 ('f433a', 'Jacquith, Elijah'),
 ('f434a', 'Jager Waffenfabik'),
 ('f435a', 'JBJ Custom Rifle'),
 ('f436a', 'JC Higgins'),
 ('f437a', 'JD Machine'),
 ('f438a', 'Jeffery, WJ & Co Ltd'),
 ('f439a', 'Jenison, J & Co.'),
 ('f440a', 'Jenks Carbine'),
 ('f441a', 'Jennings Firearms Co'),
 ('f442a', 'Jing Gong'),
 ('f443a', 'Johnson, Stan Bye & Co'),
 ('f444a', 'Joslyn'),
 ('f445a', 'Joslyn Firearms Company'),
 ('f446a', 'JP Rifles'),
 ('f447a', 'Jurras, Lee'),
 ('f448a', 'Justice,PS'),
 ('f449a', 'K.F.C'),
 ('f450a', 'KAHR'),
 ('f451a', 'Kahr Arms'),
 ('f452a', 'Kaiser Defense'),
 ('f453a', 'Kassnar Imports, Inc'),
 ('f454a', 'KBI, Inc'),
 ('f455a', 'KDF, Inc'),
 ('f456a', 'Keberst International'),
 ('f457a', 'KEL-TEC'),
 ('f458a', 'Kendall,Nicanor'),
 ('f459a', 'Keno'),
 ('f460a', 'Kessler Arms Corportation'),
 ('f461a', 'Keystone Sporting Arms'),
 ('f462a', 'Kidd'),
 ('f463a', 'Kimball Arms Company'),
 ('f464a', 'Kimber'),
 ('f465a', 'King Pin'),
 ('f466a', 'KJW (Kuan Ju Works)'),
 ('f467a', 'Klipzig & Company'),
 ('f468a', 'Knight'),
 ('f469a', 'Knight Rifles'),
 ('f470a', 'Knight´s Armament Company'),
 ('f471a', 'Korth'),
 ('f472a', 'Krieghoff'),
 ('f473a', 'Kriss'),
 ('f474a', 'KWA'),
 ('f475a', 'Lamb, HC & Co'),
 ('f476a', 'Lanber'),
 ('f477a', 'Lancaster Arms'),
 ('f478a', 'lancer'),
 ('f479a', 'LAR Manufacturing'),
 ('f480a', 'LaRue Tactical'),
 ('f481a', 'Laseraim Arms'),
 ('f482a', 'LaserLyte'),
 ('f483a', 'Lauer'),
 ('f484a', 'Laurona'),
 ('f485a', 'Law Enforcement Ordnance Corp'),
 ('f486a', 'Lawton / Border'),
 ('f487a', 'Lazzeroni'),
 ('f488a', 'Lazzeroni Arms Company'),
 ('f489a', 'LCW'),
 ('f490a', 'Leapers UTG'),
 ('f491a', 'Lee Enfield'),
 ('f492a', 'Les Baer Custom'),
 ('f493a', 'Liberty Tactical'),
 ('f494a', 'Linde A.'),
 ('f495a', 'Lionheart'),
 ('f496a', 'Ljutic'),
 ('f497a', 'Llama'),
 ('f498a', 'LMT'),
 ('f499a', 'Loki Weapons System'),
 ('f500a', 'Lone Star'),
 ('f501a', 'Lone Wolf'),
 ('f502a', 'Long Tom'),
 ('f503a', 'Lorcin Engineering Co'),
 ('f504a', 'LRB'),
 ('f505a', 'Luger'),
 ('f506a', 'LWRC'),
 ('f507a', 'M+M'),
 ('f508a', 'M96 Romanian'),
 ('f509a', 'MadBull'),
 ('f510a', 'Magnum Research'),
 ('f511a', 'Magnum Research Inc'),
 ('f512a', 'Mags'),
 ('f513a', 'Makarov'),
 ('f514a', 'Mannlicher Schoenauer'),
 ('f515a', 'Manuel Escodin'),
 ('f516a', 'Marathon Products'),
 ('f517a', 'Marksman'),
 ('f518a', 'Marksman Repeater'),
 ('f519a', 'Marlin'),
 ('f520a', 'Marlin Firearms Co.'),
 ('f521a', 'Marocchi, Armi'),
 ('f522a', 'Martin'),
 ('f523a', 'Martini Henry'),
 ('f524a', 'Marvel'),
 ('f525a', 'MAS'),
 ('f526a', 'MasterPiece Arms'),
 ('f527a', 'Maunz Mfg.,Inc'),
 ('f528a', 'Mauser'),
 ('f529a', 'Mauser Werke'),
 ('f530a', 'Mauser Werke Borsigwalde'),
 ('f531a', 'Maverick'),
 ('f532a', 'Maverick Arms, Inc'),
 ('f533a', 'McMillan & Co Inc'),
 ('f534a', 'McRee Custom'),
 ('f535a', 'Mega'),
 ('f536a', 'Mendi'),
 ('f537a', 'Mercury'),
 ('f538a', 'Merkel'),
 ('f539a', 'metro Arms'),
 ('f540a', 'MetroArms'),
 ('f541a', 'Micro Tech Small Arms'),
 ('f542a', 'Miida'),
 ('f543a', 'Miroku'),
 ('f544a', 'Mitchell Arms, Inc'),
 ('f545a', 'Mitchell´s Mausers'),
 ('f546a', 'MK Arms, Inc'),
 ('f547a', 'Morini'),
 ('f548a', 'Mosin nagant'),
 ('f549a', 'Mosin-Nagant'),
 ('f550a', 'Mosin-Nagant (D)'),
 ('f551a', 'Moss'),
 ('f552a', 'Mossberg'),
 ('f553a', 'Motiuk'),
 ('f554a', 'Mountain Rifles Inc'),
 ('f555a', 'Musgrave'),
 ('f556a', 'Nagant'),
 ('f557a', 'Navy Arms'),
 ('f558a', 'Nesika Bay Prescision, Inc'),
 ('f559a', 'New England Firearms'),
 ('f560a', 'New Haven'),
 ('f561a', 'New Ultralight Arms'),
 ('f562a', 'Next Level Training'),
 ('f563a', 'Nighthawk Custom'),
 ('f564a', 'Noble'),
 ('f565a', 'Nodak Spud'),
 ('f566a', 'Nordic Components'),
 ('f567a', 'Norinco'),
 ('f568a', 'North American Arms'),
 ('f569a', 'North Eastern Arms'),
 ('f570a', 'Nosler Custom'),
 ('f571a', 'Noveske Rifleworks'),
 ('f572a', 'Nowlin'),
 ('f573a', 'Oberland Arms'),
 ('f574a', 'ODI'),
 ('f575a', 'Ohio Ordnance'),
 ('f576a', 'Olympic Arms'),
 ('f577a', 'Omark'),
 ('f578a', 'Omega Firearms'),
 ('f579a', 'Opus Sporting Arms'),
 ('f580a', 'Oregon Arms'),
 ('f581a', 'Ortgies'),
 ('f582a', 'Own'),
 ('f583a', 'Pacific Armament Corp'),
 ('f584a', 'Page-Lewis Arms Co'),
 ('f585a', 'Palmetto State Armory'),
 ('f586a', 'Para-Ordnance'),
 ('f587a', 'Pardini'),
 ('f588a', 'Parker-Hale'),
 ('f589a', 'Patriot Ordenance'),
 ('f590a', 'PepperBall'),
 ('f591a', 'Perazzi'),
 ('f592a', 'Phoenix Arms'),
 ('f593a', 'Pietro Beretta'),
 ('f594a', 'Piotti, F. LLI'),
 ('f595a', 'Pistoolit'),
 ('f596a', 'Plainsman'),
 ('f597a', 'PolarStar'),
 ('f598a', 'Poly Technologies'),
 ('f599a', 'polytech'),
 ('f600a', 'Predator Tactical'),
 ('f601a', 'Primary Weapon Systems'),
 ('f602a', 'PSA'),
 ('f603a', 'PSE SUPRA'),
 ('f604a', 'PTR 91 Inc'),
 ('f605a', 'Puma'),
 ('f606a', 'Purdey, James'),
 ('f607a', 'PWA'),
 ('f608a', 'Quality Hardware'),
 ('f609a', 'R.A.R. Guns'),
 ('f610a', 'Radom'),
 ('f611a', 'Rainey'),
 ('f612a', 'Rainier Arms'),
 ('f613a', 'Randall Firearms'),
 ('f614a', 'Raven Arms Co'),
 ('f615a', 'Rebel Arms'),
 ('f616a', 'Red Jacket Firearms'),
 ('f617a', 'Remington'),
 ('f618a', 'Remington Custom'),
 ('f619a', 'Rexio'),
 ('f620a', 'RG Industries'),
 ('f621a', 'Rizzini, B.'),
 ('f622a', 'Robinson Arms'),
 ('f623a', 'Rock Island Armory'),
 ('f624a', 'Rock River'),
 ('f625a', 'Rocky Mountain Arms'),
 ('f626a', 'Roggio Arsenal'),
 ('f627a', 'Rohm'),
 ('f628a', 'Romanian'),
 ('f629a', 'Romanian WASR'),
 ('f630a', 'Romanian WASR (D)'),
 ('f631a', 'Rossi'),
 ('f632a', 'Rossi, Amadeo'),
 ('f633a', 'RPA'),
 ('f634a', 'RS'),
 ('f635a', 'Ruby'),
 ('f636a', 'Ruger'),
 ('f637a', 'Ruger .380'),
 ('f638a', 'Ruger Sturm'),
 ('f639a', 'RWS'),
 ('f640a', 'Saber Tactical'),
 ('f641a', 'Sabre Defense'),
 ('f642a', 'Saftey Harbor'),
 ('f643a', 'Saginaw SG'),
 ('f644a', 'Saiga'),
 ('f645a', 'Sako'),
 ('f646a', 'Sako Mosin'),
 ('f647a', 'Sar'),
 ('f648a', 'Sauer'),
 ('f649a', 'Sauer&suhl'),
 ('f650a', 'Savage Arms'),
 ('f651a', 'SCCY'),
 ('f652a', 'Schmeisser'),
 ('f653a', 'Sears'),
 ('f654a', 'Seekins Precision'),
 ('f655a', 'Sharps'),
 ('f656a', 'Shiloh Sharps'),
 ('f657a', 'Sig Arms'),
 ('f658a', 'Sig Sauer'),
 ('f659a', 'Simonov'),
 ('f660a', 'Sirt'),
 ('f661a', 'Sitima colt'),
 ('f662a', 'SKB'),
 ('f663a', 'SKS'),
 ('f664a', 'Smith & Wesson'),
 ('f665a', 'SMOS - BOHICA'),
 ('f666a', 'Socom Gear'),
 ('f667a', 'SOG Armory'),
 ('f668a', 'Special Weapons'),
 ('f669a', 'Spencer & Co.'),
 ('f670a', 'Sphinx'),
 ('f671a', 'Spikes Tactical'),
 ('f672a', 'Spoon'),
 ('f673a', 'Sportsmarketing'),
 ('f674a', 'Sportswereus/r Guns'),
 ('f675a', 'Springfield'),
 ('f676a', 'Springfield Armory'),
 ('f677a', 'SPS'),
 ('f678a', 'Squires Bingham'),
 ('f679a', 'SSP'),
 ('f680a', 'St. Etienne Chamelot Delvigne'),
 ('f681a', 'Stag Arms'),
 ('f682a', 'Stallard Arms Inc'),
 ('f683a', 'Star'),
 ('f684a', 'Sterling Arms'),
 ('f685a', 'Stevens Arms'),
 ('f686a', 'Steyr'),
 ('f687a', 'STI'),
 ('f688a', 'Stirling'),
 ('f689a', 'Stocking & Company'),
 ('f690a', 'Stoeger'),
 ('f691a', 'Sturm Ruger'),
 ('f692a', 'Sun Devil'),
 ('f693a', 'Superior Arms'),
 ('f694a', 'Surgeon'),
 ('f695a', 'SVI'),
 ('f696a', 'Tactical Innovations'),
 ('f697a', 'Tactical Rifles'),
 ('f698a', 'Tactical Solution'),
 ('f699a', 'Tactical Weapon Solutions'),
 ('f700a', 'Tanfoglio'),
 ('f701a', 'Tau Brno'),
 ('f702a', 'Taurus'),
 ('f703a', 'Tavor'),
 ('f704a', 'Taylors & Co. Inc.'),
 ('f705a', 'TDI'),
 ('f706a', 'Technoarms'),
 ('f707a', 'TGI'),
 ('f708a', 'Theoben'),
 ('f709a', 'Thompson'),
 ('f710a', 'Thompson Center'),
 ('f711a', 'Tikka'),
 ('f712a', 'Tisas'),
 ('f713a', 'Tokarev'),
 ('f714a', 'Tokyo Marui'),
 ('f715a', 'Toz'),
 ('f716a', 'Tripp Research'),
 ('f717a', 'Tristar'),
 ('f718a', 'Tula Arms'),
 ('f719a', 'Tula Arsenal'),
 ('f720a', 'Turkish Mauser'),
 ('f721a', 'Uberti'),
 ('f722a', 'Umarex'),
 ('f723a', 'Underwood'),
 ('f724a', 'Unique'),
 ('f725a', 'Unique Alpine'),
 ('f726a', 'United States Fire Arms'),
 ('f727a', 'Uronen Precision'),
 ('f728a', 'US MIA'),
 ('f729a', 'Valmet'),
 ('f730a', 'Vandal Arsenal'),
 ('f731a', 'Various'),
 ('f732a', 'Vector'),
 ('f733a', 'Vektor'),
 ('f734a', 'VEPR'),
 ('f735a', 'Volquartsen Custom'),
 ('f736a', 'VZ-24'),
 ('f737a', 'W.W. Greener'),
 ('f738a', 'Walther Arms'),
 ('f739a', 'Wards Western Field'),
 ('f740a', 'Weatherby'),
 ('f741a', 'Webley & Scott'),
 ('f742a', 'Weihrauch'),
 ('f743a', 'Wesson/Harrington'),
 ('f744a', 'Western Arms'),
 ('f745a', 'Wilson Combat'),
 ('f746a', 'Winchester'),
 ('f747a', 'Windham Weaponry'),
 ('f748a', 'WKJ'),
 ('f749a', 'WMD'),
 ('f750a', 'Xtreme Machining'),
 ('f751a', 'Yankee Hill Machine'),
 ('f752a', 'Zastava'),
 ('f753a', 'Zel Manufacturing'),
 ('f754a', 'ZMZ'),
 ('f755a', 'Custom Rifle'),
 ('f756a', 'Custom Revolver'),
 ('f757a', 'Custom Pistol'),
 ('f758a', 'Custom Shotgun'),
 ('f759a', 'Custom Air'),
 ('f760a', 'Custom Muzzle loader'),
 ('f761a', 'Custom SMG'),
 ('f762a', 'Unknown'),
 ('f1a', 'Rental'),
)

SIGHT\_MAKE = (
 ('siXX', 'Unknown'),
 ('si2a', 'Aimpoint'),
 ('si3a', 'Alpen'),
 ('si4a', 'Armasight'),
 ('si5a', 'Barska'),
 ('si6a', 'Beretta'),
 ('si7a', 'Bobro'),
 ('si8a', 'Browning'),
 ('si9a', 'BSA'),
 ('si10a', 'Burris'),
 ('si11a', 'Bushnell'),
 ('si12a', 'C-more'),
 ('si13a', 'Crimson Trace'),
 ('si14a', 'Elcan'),
 ('si15a', 'EOTech'),
 ('si16a', 'FNH'),
 ('si17a', 'G&G'),
 ('si18a', 'Hawke'),
 ('si19a', 'Hi-Lux'),
 ('si20a', 'JP'),
 ('si21a', 'Kahles'),
 ('si22a', 'Laser Devices'),
 ('si23a', 'Laserlyte'),
 ('si24a', 'Lasermax'),
 ('si25a', 'Leupold'),
 ('si26a', 'Lucid Optics'),
 ('si26b', 'Meopta'),
 ('si27a', 'Meprolight'),
 ('si28a', 'Millet'),
 ('si29a', 'Mueller'),
 ('si30a', 'ncStar'),
 ('si31a', 'Nikon'),
 ('si32a', 'Nightforce'),
 ('si33a', 'Redfield'),
 ('si34a', 'Ruger'),
 ('si35a', 'Schmidt Bender'),
 ('si36a', 'Sightmark'),
 ('si37a', 'Steiner'),
 ('si38a', 'Sightron'),
 ('si39a', 'Simmons'),
 ('si40a', 'Swarowski'),
 ('si41a', 'Trijicon'),
 ('si42a', 'Truglo'),
 ('si43a', 'US Optics'),
 ('si44a', 'Viridian'),
 ('si45a', 'Vortex'),
 ('si46a', 'Zeiss'),
)

FIREARM\_TYPE = (
 ('hg', 'Handgun'),
 ('rf', 'Rifle'),
 ('sg', 'Shotgun'),
 ('sm', 'SMG'),
 ('mz', 'Muzzle Loader'),
 ('ag', 'Air Handgun'),
 ('ar', 'Air Rifle'),
 ('hc', 'Handgun conversion kit'),
 ('rc', 'Rifle converstion kit'),
 ('sc', 'Shotgun converstion kit'),
)

FIREARM\_USAGE = (
 ('ht', 'Hunting'),
 ('tr', 'Training'),
 ('cp', 'Competition'),
 ('wc', 'Duty/Work'),
 ('rt', 'Rental'),
 ('ct', 'Collectible'),
)

FIREARM\_CLASS = (
 ('ow', 'Own'),
 ('rt', 'Rental'),
 ('sg', 'Issued'),
 ('bw', 'Borrowed'),
 ('sl', 'Sold'),
)

API\_FIREARM\_FIELDS = API\_PREMIUM\_FIELDS + (
 'input\_units',
 'output\_units',
 'weapon\_type',
 'weapon\_class',
 'weapon\_usage',
 'make',
 'name',
 'caliber',
 'serial',
 'barrel\_twist',
 'barrelt\_twist\_direct',
 'rounds\_fired',
)

API\_FIREARM\_EXTRAS = API\_PREMIUM\_EXTRAS + (
 'barrel\_length',
 'purchased',
 'built',
 'last\_fired',
 'last\_cleaned',
 'last\_repaired',
)

API\_FIREARM\_EXCLUDES = API\_PREMIUM\_EXCLUDES + (
 'input\_units',
 'output\_units',
 'rounds\_fired',
 'purchased',
 'serial',
 'built',
 'last\_fired',
 'last\_cleaned',
 'last\_repaired',
)

class Firearm(PremiumModel):

 input\_units = models.CharField(\_(u'Units'), max\_length=3, choices=INPUT\_UNITS, default='met')
 output\_units = models.CharField(\_(u'Out Units'), max\_length=8, choices=OUTPUT\_UNITS, default='MOA')

 weapon\_type = models.CharField(\_(u'Type'), max\_length=3, choices=FIREARM\_TYPE, default='hg')
 weapon\_class = models.CharField(\_(u'Class'), max\_length=3, choices=FIREARM\_CLASS, default='ow')

 #weapon\_usage = models.CharField(\_(u'Main usage'), max\_length=3, choices=FIREARM\_USAGE, default='tr')
 weapons\_usage = MultiSelectField(\_(u'Usage'), max\_length=400, choices=FIREARM\_USAGE)

 make = models.CharField(\_(u'Brand'), max\_length=8, choices=FIREARMS\_MAKE, default='fi0', db\_index=True)
 name = models.CharField(\_(u'Model'), max\_length=40, db\_index=True)
 caliber = models.CharField(\_(u'Caliber'), max\_length=8, choices=FIREARMS\_CALIBERS, default='223', db\_index=True)
 serial = models.CharField(\_(u'Serial Id.'), max\_length=300, blank=True)

 # barrel
 barrel\_twist = models.DecimalField(\_(u'Barell twist (1:n inch)'), max\_digits=5, decimal\_places=2, default=7) # in turns of inches
 barrel\_twist\_direct = models.CharField(\_(u'Twist direct'), max\_length=3, choices=TWIST, default='R') # direction of twist

 # usage
 rounds\_fired = models.PositiveIntegerField(\_('Rounds fired'), default=0) # will auto-increment with usage in SSI

Fields in use

Make Optional
name Mandatory
caliber Optional
weapon\_type Mandatory
barrel\_length cm/inch/cm (two decimals) Optional
serial Optional
rounds\_fired Optional
purchased\_date yyyy-mm-dd Optional
comment Optional
input\_units Mandatory
visibility Mandatory

## Sight

Model, fields, extras and excluded

TURRET\_UNITS = (
 ('X', 'not set'),
 ('2', '2'),
 ('1', '1'),
 ('1/2', '1/2'),
 ('1/3', '1/3'),
 ('1/4', '1/4'),
 ('1/5', '1/5'),
 ('1/6', '1/6'),
 ('1/8', '1/8'),
 ('5/8', '5/8'),
 ('1/10', '1/10'),
 ('1/20', '1/20'),
)

TURRET\_SCALES = (
 ('X', 'not set'),
 ('CPHM', 'Centimeter per hundred meters (CPHM)'),
 ('MOA', 'Minute of Angle (MOA)'),
 ('IPHY', 'Inch per hundred yards (IPHY)'),
 ('IPHM', 'Inch per hundred meters (IPHM)'),
 ('MILS', 'MILS'),
 ('MDot', 'MDot (6400 mills/circle)'),
 ('MRad', 'MRad (true Milliradian as used by USMC)'),
)

"""
 CPHM (Centimeter Per Hundred Meters): If range is in meters then CPHM makes things simple if working in the Metric system. A change of 1 CPHM changes the impact point of a bullet by 1 centimeter at 100 meters, 2 centimeters at 200 meters, 3 centimeters at 300 meters and so on. CPHM divides a full circle into 62,832 parts. CPHM is also 0.1 milliradian.

 IPHM (Inch Per Hundred Meters): If range is in meters as it is in some competitions, then IPHM makes things simple. A change of 1 IPHM changes the impact point of a bullet by 1 inch at 100 meters, 2 inches at 200 meters, 3 inches at 300 meters and so on. IPHM divides a full circle into 24,737 parts.

 IPHY (Inch Per Hundred Yards): Shooters often round off the MOA value of 1.047 inches per hundred yards to 1 inch, but that starts to create a significant error at longer ranges. Thus, IPHY was invented and is sometimes called the shooter's MOA or SMOA. A change of 1 IPHY changes the impact point of a bullet by 1 inch at 100 yards, 2 inches at 200 yards, 3 inches at 300 yards and so on. IPHY divides a full circle into 22,619 parts.

 MOA (Minute Of Angle) is a standard unit of angle used in science and engineering. A circle is divided into 360 degrees and each degree is divided into 60 minutes for a total of 21,600 minutes of angle in a circle. Most optical sights are calibrated in MOA often with 2, 4 or 8 clicks per MOA. A change of 1 MOA changes the impact point of a bullet by 1.047 inches at 100 yards, 2.094 inches at 200 yards, 3.141 inches at 300 yards and so on.

 MDot (6400 mils/circle used by Army) is a modified version of Milliradian. Where Milliradian divides a full circle into 6,283 parts MDot divides a full circle into 6,400 parts. The Army did this to make it easier to work with standard Army maps while introducing only a 1.8% error in sight adjustment changes.

 MRad (true Milliradian used by USMC) is a standard unit of angle used in science and engineering and is gaining popularity with hunters and shooters. A radian is the angle produced by marking off the distance of a circle's radius along it's circumference. A milliradian is that angle divided by 1,000. In practical terms the radius is the distance from the gun to the target and a change of 1 milliradian (MRad) changes the impact point of a bullet by 1/1,000 the distance to the target. If the target is 1,000 feet away a change of 1 MRad changes the impact point of a bullet by 1 foot. If the target is 1,000 yards away a change of 1 MRad changes the impact point of a bullet by 1 yard. If the target is 1,000 meters away a change of 1 MRad changes the impact point of a bullet by 1 meter. If the target is 500 meters away a change of 1 MRad changes the impact point of a bullet by 1/2 meter. The exact 1 to 1,000 ratio between impact change and range to the target in whatever units you use for range makes doing the math in your head relatively simple. MRad divides a full circle into 6,283 parts.

"""

SIGHT\_TYPE = (
 ('fct', 'Factory'),
 ('iro', 'Iron Sights'),
 ('bui', 'Back-up Iron Sights'),
 ('sco', 'Scope'),
 ('red', 'Red dot'),
 ('swr', 'Scope w. red-dot'),
 ('hol', 'Holographic'),
 ('lzr', 'Laser sight'),
 ('nvs', 'Night-vision scope'),
)

DEFAULT\_RETICLES = (
 ('xxxxx', 'Not set'),

-- THIS WILL BE ADDED AS SOON AS WE FIND A RETICLE LISTING --
)

IRON\_RETICLES = (-- THIS WILL BE ADDED AS SOON AS WE FIND A RETICLE LISTING --)

SCOPE\_RETICLES = (-- THIS WILL BE ADDED AS SOON AS WE FIND A RETICLE LISTING --)

LASER\_RETICLES = (
 ('g', 'Green'),
 ('r', 'Red'),
)

REDDOT\_RETICLES = (
 ('1', '1 MOA'),
 ('2', '2 MOA'),
 ('4', '4 MOA'),
 ('8', '8 MOA'),
)

RETICLES = IRON\_RETICLES + SCOPE\_RETICLES + REDDOT\_RETICLES + LASER\_RETICLES

API\_SIGHT\_FIELDS = API\_PREMIUM\_FIELDS + (
 'input\_units',
 'output\_units',
 'sight\_type',
 'make',
 'model',
 'serial',
# 'reticle',
 'firearm',
)

API\_SIGHT\_EXTRAS = API\_PREMIUM\_EXTRAS + (
 'purchased',
 'height',
 'offset',
 'zeroed\_distance',
 'zeroed\_shift\_elevation',
 'zeroed\_shift\_windage',
)

API\_SIGHT\_EXCLUDES = API\_PREMIUM\_EXCLUDES + (
 'input\_units',
 'output\_units',
 'serial',
 'purchased',
 'height',
 'offset',
 'zeroed\_distance',
 'zeroed\_shift\_elevation',
 'zeroed\_shift\_windage',
)

class Sight(PremiumModel):

 input\_units = models.CharField(\_(u'Units'), max\_length=3, choices=INPUT\_UNITS, default='met')
 output\_units = models.CharField(\_(u'Out Units'), max\_length=8, choices=OUTPUT\_UNITS, default='MOA')

 # general info
 sight\_type = models.CharField(\_(u'Sight type'), max\_length=5, choices=SIGHT\_TYPE, default='sco')
 make = models.CharField(\_(u'Brand'), max\_length=6, choices=SIGHT\_MAKE, default='siXX', db\_index=True)
 model = models.CharField(\_(u'Modell'), max\_length=40, blank=True)
 serial = models.CharField(\_(u'Serial Id.'), max\_length=300, blank=True)
 reticle = models.CharField(\_(u'Reticle'), max\_length=5, choices=RETICLES, default='xxxxx') # to be defined later

 elevation\_unit = models.CharField(\_(u'Elevation unit'), max\_length=5, choices=TURRET\_SCALES, default='MOA') # unit of clicks
 elevation\_turret\_grad = models.CharField(\_(u'Elevation turret grad (click)'), max\_length=5, choices=TURRET\_UNITS, default='1/4') # size of click
 windage\_unit = models.CharField(\_(u'Windage unit'), max\_length=5, choices=TURRET\_SCALES, default='1/4') # unit of clicks
 windage\_turret\_grad = models.CharField(\_(u'Windage turret grad'), max\_length=5, choices=TURRET\_UNITS, default='MOA') # size of click
 lead\_unit = models.CharField(\_(u'Lead unit'), max\_length=5, choices=TURRET\_SCALES, default='MOA') # prefered lead unit of moving target

 # details on where it is in use and how it is sighted in for this firearm
 firearm = models.ForeignKey(Firearm, verbose\_name=\_(u'In use on'), blank=True, null=True)

### Fields in use

sight\_type Mandatory
make Mandatory
model Mandatory
serial Optional
firearm pk to firearm Optional
height mm/inch/mm (two decimal) Optional
offset mm/inch/mm (two decimal) Optional
zeroed\_distance m/yards/m (integer) Optional
zeroed\_shift\_elevation mm/inch/mm (two decimal) Optional
zeroed\_shift\_windage mm/inch/mm (two decimal) Optional
purchased\_date yyyy-mm-dd Optional
comment Optional
input\_units Mandatory
visibility Mandatory

## Ammunition

Model, fields, extras and excluded

AMMO\_TYPE = (
 ('hld', u'Handloaded'),
 ('fct', u'Factory'),
)

BC\_SCALE = (
 ('G1', u'G1'),
 ('G7', u'G7'),
)

API\_AMMUNITION\_FIELDS = API\_PREMIUM\_FIELDS + (
 'recipe',
 'input\_units',
 'output\_units',
 'ammo\_type',
 'batch\_name',
 'caliber',
 'purpose',
 'take\_caution',
 'rounds',
 'rounds\_fired',
 'powder\_make',
 'powder\_lot',
 'bullet\_make',
 'bullet\_bc',
 'bullet\_bc\_drag\_model',
 'bullet\_cost',
 'primer\_make',
 'primer\_cost',
 'brass\_make',
 'brass\_cost',
 'other\_cost\_per\_round',
 'other\_total\_cost',
 'currency',
)

API\_AMMUNITION\_EXTRAS = API\_PREMIUM\_EXTRAS + (
 'oal',
 'powder\_weight\_gram',
 'powder\_cost',
 'bullet\_weight\_gram',
 'bullet\_lenght\_mm',
 'bullet\_diameter\_mm',
)

API\_AMMUNITION\_EXCLUDES = API\_PREMIUM\_EXCLUDES + (
 'input\_units',
 'output\_units',
 'rounds',
 'rounds\_fired',
 'powder\_lot',
 'powder\_cost',
 'bullet\_cost',
 'primer\_cost',
 'brass\_cost',
 'other\_cost\_per\_round',
 'other\_total\_cost',
)

class Ammunition(PremiumModel):
 #
 # '''
 # Ammunition represents the actual rounds/ammo. Then depending if it is factory or handloaded/reloaded it
 # will reference a handload recipe or not.
 #
 # If it is handloaded then all 'data' from the handload recipe is copied over to this specific instance of ammunition
 # as a way to express how it looks just now.
 #
 # '''

 recipe = models.ForeignKey(HandloadRecipe, verbose\_name=\_(u'Handload recipe'), blank=True, null=True) # is this handload, if not set then factory

 input\_units = models.CharField(\_(u'Units'), max\_length=3, choices=INPUT\_UNITS, default='met')
 output\_units = models.CharField(\_(u'Out Units'), max\_length=8, choices=OUTPUT\_UNITS, default='MOA')
 ammo\_type = models.CharField(\_(u'Type'), max\_length=3, choices=AMMO\_TYPE, default='fct', db\_index=True)
 batch\_name = models.CharField(\_(u'Batch name/id'), max\_length=40, db\_index=True)
 caliber = models.CharField(\_(u'Caliber'), max\_length=8, choices=FIREARMS\_CALIBERS, default='223', db\_index=True)
 purpose = MultiSelectField(\_(u'Purpose'), max\_length=400, choices=AMMO\_PURPOSE)
 take\_caution = models.BooleanField(\_("Take caution, special usage, requirements or other"), default=False)

 #
 # Values below are not editable
 #

 # OAL = overall length, COL is SAAMI max cartridge overall length so we use OAL as our ammo specific length for load
 \_oal\_mm = models.DecimalField(\_(u'Overall length (mm)'), max\_digits=10, decimal\_places=3, default=0) # in mm, show mm w. 1 decimal and inches w. 2 decimals
 rounds = models.IntegerField(\_('#Rounds'), default=0)
 rounds\_fired = models.IntegerField(\_('#Rounds fired'), default=0)

 powder\_make = models.CharField(\_(u'Powder brand'), max\_length=40, blank=True, db\_index=True)
 powder\_lot = models.CharField(\_(u'Powder batch'), max\_length=40, blank=True, db\_index=True)
 \_powder\_weight\_gram = models.DecimalField(\_(u'Powder weight (g)'), max\_digits=12, decimal\_places=3, default=0)
 \_powder\_cost\_500\_gram = models.DecimalField(\_(u'Powder cost (500g)'), max\_digits=20, decimal\_places=3, default=0)
 bullet\_make = models.CharField(\_(u'Bullet'), max\_length=40, blank=True, db\_index=True)
 \_bullet\_weight\_gram = models.DecimalField(\_(u'Bullet weight (g)'), max\_digits=10, decimal\_places=3, default=0)
 \_bullet\_length\_mm = models.DecimalField(\_(u'Bullet length (mm)'), max\_digits=10, decimal\_places=4, default=0)
 \_bullet\_diameter\_mm = models.DecimalField(\_(u'Bullet diameter (mm)'), max\_digits=10, decimal\_places=4, default=0)
 bullet\_bc = models.DecimalField(\_(u'Ballistic Coefficient'), max\_digits=10, decimal\_places=3, default=0)
 bullet\_bc\_drag\_model = models.CharField(\_(u'BC Drag model'), max\_length=3, choices=BC\_SCALE, default='G1')
 bullet\_cost = models.DecimalField(\_(u'Proj cost'), max\_digits=10, decimal\_places=2, default=0)
 primer\_make = models.CharField(\_(u'Primer make'), max\_length=40, blank=True, db\_index=True)
 primer\_cost = models.DecimalField(\_(u'Primer cost'), max\_digits=10, decimal\_places=2, default=0)
 brass\_make = models.CharField(\_(u'Brass make'), max\_length=40, blank=True, db\_index=True)
 brass\_cost = models.DecimalField(\_(u'Brass cost'), max\_digits=10, decimal\_places=2, default=0)
 other\_cost\_per\_round = models.DecimalField(\_(u'Other cost per round'), max\_digits=10, decimal\_places=2, default=0)
 other\_total\_cost = models.DecimalField(\_(u'Other total costs'), max\_digits=10, decimal\_places=2, default=0)
 currency = models.CharField(\_(u'Currency'), max\_length=3, choices=CURRENCIES\_ISO\_4217, default='EUR')

Fields - HandloadED Ammunition

The following fields are used for reading/creating handloaded ammunition:

recipe pk to recipe Optional
batch\_name Optional
rounds Mandatory
purpose Optional
comment Optional
visibility Mandatory

(several other information like caliber etc. are copied from recipe etc. to the new ammunition, this is the reason for this short list).

Fields - Factory Ammunition

The following fields are used for reading/creating factory/stock ammunition

batch\_name Optional
caliber Mandatory
bullet\_make Optional
rounds Mandatory
rounds\_fired Optional
bullet\_weight gram/grains/grains (two decimals) Optional
bullet\_length mm/inch/mm (two decimals) Optional
bullet\_diameter mm/inch/mm (two decimals) Optional
bullet\_bc Optional
bullet\_bc\_drag\_model Optional
purpose Optional
comment Optional
input\_units Mandatory
visibility Mandatory

## Handload Recipe

A Handload Recipe is for loading ammo. Note that SSI takes no responsibility for whatever information you enter or share over this API XXX

### Model, fields, extras and excluded

BC\_SCALE = (
 ('G1', u'G1'),
 ('G7', u'G7'),
)

AMMO\_PURPOSE = (
 ('tr', u'Training'),
 ('co', u'Competition'),
 ('pr', u'Precision'),
 ('hu', u'Hunting'),
 ('dt', u'Duty'),
 ('ip', u'IPSC'),
 ('up', u'USPSA'),
 ('id', u'IDPA'),
 ('pp', u'PPC'),
 ('sr', u'SRA'),
 ('ir', u'ICORE'),
 ('ss', u'SASS'),
 ('sc', u'Steel'),
 ('ns', u'NSSF Rimfire'),
 ('nd', u'Nordic'),
)

API\_HANDLOAD\_FIELDS = API\_PREMIUM\_FIELDS + (
 'input\_units',
 'output\_units',
 'name',
 'caliber',
 'purpose',
 'take\_caution',
 'powder\_make',
 'powder\_lot',
 'bullet\_make',
 'bullet\_bc',
 'bullet\_bc\_drag\_model',
 'bullet\_cost',
 'primer\_make',
 'primer\_cost',
 'brass\_make',
 'brass\_cost',
 'other\_cost\_per\_round',
 'other\_total\_cost',
 'currency', )

API\_HANDLOAD\_EXTRAS = API\_PREMIUM\_EXTRAS + (
 'oal'
 'powder\_weight',
 'powder\_cost',
 'bullet\_weight',
 'bullet\_lenght',
 'bullet\_diameter',
)

API\_HANDLOAD\_EXCLUDES = API\_PREMIUM\_EXCLUDES + (
 'input\_units',
 'output\_units',
 'rounds',
 'rounds\_fired',
 'powder\_lot',
 'powder\_cost',
 'bullet\_cost',
 'primer\_cost',
 'brass\_cost',
 'other\_cost\_per\_round',
 'other\_total\_cost',
 'currency',
)

class HandloadRecipe(PremiumModel):

 input\_units = models.CharField(\_(u'Units'), max\_length=3, choices=INPUT\_UNITS, default='met')
 output\_units = models.CharField(\_(u'Out Units'), max\_length=8, choices=OUTPUT\_UNITS, default='MOA')

 name = models.CharField(\_(u'Name/reference'), max\_length=40, db\_index=True)
 caliber = models.CharField(\_(u'Caliber'), max\_length=8, choices=FIREARMS\_CALIBERS, default='223', db\_index=True)
 purpose = MultiSelectField(\_(u'Purpose'), max\_length=400, choices=AMMO\_PURPOSE)
 take\_caution = models.BooleanField(\_("Take caution, special usage, requirements or other"), default=False)

 # OAL = overall length, COL is SAAMI max cartridge overall length so we use OAL as our ammo specific length for load
 powder\_make = models.CharField(\_(u'Powder'), max\_length=40, blank=True, db\_index=True)
 powder\_lot = models.CharField(\_(u'Powder batch'), max\_length=40, blank=True, db\_index=True)

 bullet\_make = models.CharField(\_(u'Bullet'), max\_length=40, blank=True, db\_index=True)
 bullet\_bc = models.DecimalField(\_(u'Ballistic Coefficient'), max\_digits=10, decimal\_places=3, default=0)
 bullet\_bc\_drag\_model = models.CharField(\_(u'BC Drag model'), max\_length=3, choices=BC\_SCALE, default='G1')
 bullet\_cost = models.DecimalField(\_(u'Bullet cost'), max\_digits=10, decimal\_places=2, default=0)
 other\_cost\_per\_round = models.DecimalField(\_(u'Other cost per round'), max\_digits=10, decimal\_places=2, default=0)

 primer\_make = models.CharField(\_(u'Primer'), max\_length=40, blank=True, db\_index=True)
 primer\_cost = models.DecimalField(\_(u'Primer cost'), max\_digits=10, decimal\_places=2, default=0)

 brass\_make = models.CharField(\_(u'Brass'), max\_length=40, blank=True, db\_index=True)
 brass\_cost = models.DecimalField(\_(u'Brass cost'), max\_digits=10, decimal\_places=2, default=0)

 currency = models.CharField(\_(u'Currency'), max\_length=3, choices=CURRENCIES\_ISO\_4217, default='EUR')

Fields in use

name Mandatory

caliber Mandatory

oal mm/inch/mm (two decimals) Optional

powder\_make Mandatory

powder\_lot Optional

powder\_weight gram/grains/grains (two decimals) Optional

powder\_cost 500g/pound/500g Optional

bullet\_make Optional

bullet\_weight gram/grains/grains (two decimals) Optional

bullet\_length mm/inch/mm (two decimals) Optional

bullet\_diameter mm/inch/mm (two decimals) Optional

bullet\_cost Optional

brass\_make Optional

brass\_cost Optional

primer\_make Optional

primer\_cost Optional

bullet\_bc Optional

bullet\_bc\_drag\_model Optional

other\_cost\_per\_round Optional

purpose Optional

comment Optional

take\_caution Mandatory

currency Mandatory

input\_units Mandatory

visibility Mandatory

# IPSC/USPSA Matches

IPSC and USPSA are supported under same API. You need to interpret and show choices depending on if match/event is USPSA or IPSC. The field definitions show you the exact choices that are in use and allowed.

## URLs

/ipsc/match/<key>/all/ [GET]

/ipsc/match/<key>/stats/ [GET]

/ipsc/match/<key>/ [GET]

/ipsc/stage/<key>/ [GET]

/ipsc/competitor/<key>/ [GET]

/ipsc/competitor/<key>/all/ [GET]

/ipsc/competitor/<key>/set\_pin/ [POST]

/ipsc/competitor/<key>/reset\_pin/ [GET]

/ipsc/competitor/<key>/verify\_all/ [POST]

/ipsc/scoresheet/<key>/ [GET DELETE]

/ipsc/scoresheet/<key>/verify/ [POST]

/ipsc/scoresheet/<key>/unverify/ [GET PUT POST]

/ipsc/squad/<key>/ [GET]

/ipsc/team/<key>/ [GET]

/ipsc/match/<key>/competitors/ [GET]

/ipsc/match/<key>/stages/ [GET]

/ipsc/match/<key>/squads/ [GET]

/ipsc/match/<key>/teams/ [GET]

/ipsc/match/<key>/scoresheets/ [POST GET PUT]

/ipsc/stage/<key>/scoresheets/ [GET]

/ipsc/competitor/<key>/scoresheets [GET]

/ipsc/match/<key>/winmss/ [GET] returns WinMSS.zip export file, see web for more info

/ipsc/match/<key>/ews/ [GET] returns EzWinScore export file, see web for more info

/ipsc/match/<key>/ced/ [GET] returns CED export file, see web for more info

Note <key> in all above expresses the primary key of the instance accessed.

## Creating or updating score sheets

Each POST or PUT, i.e. create or update of a score sheet must contain all the fields below. PK shall only be expressed in the case you are updating an existing score sheet, if creating a new ensure you provide pk for stage and competitor.

The array can contain one or more score sheets – if you post/put multiple then all will be checked and if and only if all are ok will they be updated or created.

 [

 {

 "pk": 1,

 "model": "match\_ipsc.ipscscorecard",

 "competitor": 2,

 "stage": 4

 "time": "4.21",

 "ascore": 3,

 "bscore": 0,

 "cscore": 4,

 "dscore": 1,

 "zeroed": false,

 "penalty": 1,

 "deductions": 10,

 "spec\_penalty": 0,

 "dq\_reason": "no",

 "procedural": 0,

 "miss": 0,

 "comment": "",

 },

 {

 "pk": 2,

 "model": "match\_ipsc.ipscscorecard",

 ...

 }

]

POST or PUT for scorecard must contain the following fields and these are the only ones that will be considered; 'competitor', 'time', 'ascore', 'bscore', 'cscore', 'dscore', 'miss', 'penalty', 'procedural', 'evt', 'pf\_correction', 'dq\_reason', 'comment', 'spec\_penalty', 'zeroed', 'tnn', 'tnh', 'tne', 'stage\_not\_fired'.

If any field is not applicable then set to default value.

### Validation

The following are cross-field validations that will occur on server to validate field interaction. Might be good to check before posting.

* The fields shown and used differs if stage is Comstock, Fixed Time, Virginia Count or Time-plus (need to understand these different rules, Time-plus only for USPSA Grand Tournaments)
* B is only used in USPSA, not in IPSC. Set to 0 when not used
* pf\_correction only shown/used if stage can have this
* evt10…40 (enhanced value targets) only shown/used if USPSA and Grand Tournament and stage has evt for this value set, else set to 0
* Time – time is required, unless stage is marked as DQ or Zeroed
* A + B + C + D + M must be in range of minimum\_rounds and maximum\_rounds for stage

Time-plus scoring for USPSA uses very special fields compared to that within USPSA/IPSC when scoring a stage under Comstock, Virginia-count, fixed time.

## Model, fields, extras and excluded

### Ipsc Match 🡨Match

ALL\_MATCH\_LEVELS = (('l1', u'Level I'), ('l2', u'Level II'), ('l3', u'Level III'), ('l4', u'Level IV'), ('l5', u'Level V'), ('nt', u'Nationals'))

IPSC\_MATCH\_LEVEL = (('l1', u'Level I'), ('l2', u'Level II'), ('l3', u'Level III'), ('l4', u'Level IV'), ('l5', u'Level V'))

USPSA\_MATCH\_LEVEL = (('l1', u'Level I'), ('l2', u'Level II'), ('l3', u'Level III'), ('nt', u'Nationals'))

ALL\_FIREARMS = (('hg', u'Handgun'), ('rf', u'Rifle'), ('mr', u'Mini Rifle'), ('pr', u'Precision Rifle'), ('sg', u'Shotgun'), ('ai', u'Air'))

IPSC\_FIREARM = (('hg', u'Handgun'), ('rf', u'Rifle'), ('mr', u'Mini Rifle'), ('sg', u'Shotgun'), ('ai', u'Air'))

USPSA\_FIREARM = (('hg', u'Handgun'), ('rf', u'Rifle'), ('pr', u'Precision Rifle'), ('sg', u'Shotgun'))

ALL\_FIREARM\_DIVISIONS = (('hg1', u'Open'),

 ('hg2', u'Standard'),

 ('hg3', u'Production'),

 ('hg4', u'Modified'),

 ('hg5', u'Revolver'),

 ('hg6', u'Open'),

 ('hg7', u'Production'),

 ('hg8', u'Single-Stack'),

 ('hg9', u'Limited'),

 ('hg10', u'Limited-10'),

 ('hg11', u'Revolver'),

 ('hg12', u'Classic'), # old single stack in IPSC

 ('hgc', u'Custom'),

 ('rf1', u'Semi-Auto Open'),

 ('rf2', u'Semi-Auto Standard'),

 ('rf3', u'Manual Action Open'),

 ('rf4', u'Manual Action Standard'),

 ('rf5', u'Open'),

 ('rf6', u'Standard'),

 ('rf7', u'Tactical'),

 ('rf8', u'Manually Operated'),

 ('rf9', u'Semi-Auto'),

 ('rf10', u'Manually Operated'),

 ('rf11', u'Manual Action Standard 10'),

 ('rfc', u'Custom'),

 ('mr1', u'Mini Rifle Open'),

 ('mr2', u'Mini Rifle Standard'),

 ('mrc', u'Mini Rifle Custom'),

 ('sg1', u'Open'),

 ('sg2', u'Modified'),

 ('sg3', u'Standard'),

 ('sg4', u'Standard Manual'),

 ('sg5', u'Open'),

 ('sg6', u'Limited/Tactical'),

 ('sg7', u'Heavy Metal'),

 ('sgc', u'Custom'),

 ('ai1', u'Open'),

 ('ai2', u'Standard'),

 ('ai3', u'Production'),

 ('ai4', u'Open'),

 ('ai5', u'Standard'),

 ('ai6', u'Production'),

 ('aic', u'Custom'))

IPSC\_FIREARM\_HANDGUN\_DIVISION = ( ('hg1', u'Open'),

 ('hg2', u'Standard'),

 ('hg3', u'Production'),

 ('hg5', u'Revolver'),

 ('hg12', u'Classic'),

 ('hgc', u'Custom'))

 # modified has been removed 2012 rules -- ('hg4', u'Modified')

USPSA\_FIREARM\_HANDGUN\_DIVISION = ( ('hg6', u'Open'),

 ('hg7', u'Production'),

 ('hg8', u'Single-Stack'),

 ('hg9', u'Limited'),

 ('hg10', u'Limited-10'),

 ('hg11', u'Revolver'),

 ('hgc', u'Custom'))

IPSC\_FIREARM\_RIFLE\_DIVISION = ( ('rf1', u'Semi-Auto Open'),

 ('rf2', u'Semi-Auto Standard'),

 ('rf3', u'Manual Action Open'),

 ('rf4', u'Manual Action Standard'),

 ('rf11', u'Manual Action Standard 10'), # added on test 2012-2014

 ('rfc', u'Custom'),

 )

USPSA\_FIREARM\_RIFLE\_DIVISION = (('rf5', u'Open'),

 ('rf6', u'Limited'),

 ('rf7', u'Tactical'),

 ('rf8', u'Manually Operated'),

 ('rfc', u'Custom'),

 )

USPSA\_FIREARM\_PRECISION\_RIFLE\_DIVISION = (('rf9', u'Semi-Auto'),

 ('rf10', u'Manually Operated'))

# IPSC\_FIREARM\_PRECISION\_RIFLE\_DIVISION = THERE IS NO SUCH...

IPSC\_FIREARM\_MINI\_RIFLE\_DIVISION = (('mr1', u'Mini Rifle Open'),

 ('mr2', u'Mini Rifle Standard'),

 ('mrc', u'Mini Rifle Custom'),

 )

IPSC\_FIREARM\_SHOTGUN\_DIVISION = (('sg1', u'Open'),

 ('sg2', u'Modified'),

 ('sg3', u'Standard'),

 ('sg4', u'Standard Manual'),

 ('sgc', u'Custom'))

USPSA\_FIREARM\_SHOTGUN\_DIVISION = ( ('sg5', u'Open'),

 ('sg6', u'Limited/Tactical'),

 ('sg7', u'Heavy Metal'),

 ('sgc', u'Custom'))

IPSC\_FIREARM\_AIR\_DIVISION = (('ai1', u'Open'),

 ('ai2', u'Standard'),

 ('ai3', u'Production'),

 ('aic', u'Custom'))

USPSA\_FIREARM\_AIR\_DIVISION = ( ('ai4', u'Open'),

 ('ai5', u'Standard'),

 ('ai6', u'Production'),

 ('aic', u'Custom'))

ALL\_SERIE\_DIVISIONS = (('iop', u'Open'),

 ('imd', u'Modified'),

 ('ist', u'Standard'),

 ('ipr', u'Production'),

 ('uop', u'Open'),

 ('umd', u'Limited'),

 ('ust', u'Tactical'),

 ('upr', u'Heavy Metal'),

 ('ir1', u'Restricted 1'),

 ('ir2', u'Restricted 2'),

 ('ir3', u'Restricted 3'),

 ('ir4', u'Restricted 4'),

 ('ir5', u'Restricted 5'),

 ('ur1', u'Restricted 1'),

 ('ur2', u'Restricted 2'),

 ('ur3', u'Restricted 3'),

 ('ur4', u'Restricted 4'),

 ('ur5', u'Restricted 5'))

IPSC\_SERIE\_DIVISIONS = (('iop', u'Open'),

 ('imd', u'Modified'),

 ('ist', u'Standard'),

 ('ipr', u'Production'),

 ('ir1', u'Restricted 1'),

 ('ir2', u'Restricted 2'),

 ('ir3', u'Restricted 3'),

 ('ir4', u'Restricted 4'),

 ('ir5', u'Restricted 5'))

USPSA\_SERIE\_DIVISIONS = (('uop', u'Open'),

 ('umd', u'Limited'),

 ('ust', u'Tactical'),

 ('upr', u'Heavy Metal'),

 ('ur1', u'Restricted 1'),

 ('ur2', u'Restricted 2'),

 ('ur3', u'Restricted 3'),

 ('ur4', u'Restricted 4'),

 ('ur5', u'Restricted 5'))

IPSC\_POWERFACTOR = (('+', u'Major'), ('-', u'Minor'))

ALL\_CATEGORIES = (('-', u'None'),

 ('S', u'Senior'),

 ('L', u'Lady'),

 ('J', u'Junior'),

 ('SS', u'Super Senior'),

 ('MI', u'Military'),

 ('LE', u'Law Enforcement'))

IPSC\_CATEGORY = (('-', u'None'),

 ('L', u'Lady'),

 ('J', u'Junior'),

 ('S', u'Senior'),

 ('SS', u'Super Senior'))

USPSA\_CATEGORY = (('-', u'None'),

 ('L', u'Lady'),

 ('J', u'Junior'),

 ('S', u'Senior'),

 ('SS', u'Super Senior'),

 ('MI', u'Military'),

 ('LE', u'Law Enforcement'))

IPSC\_USPSA\_CLASSIFICATION = (('U', u'Unclassified'),

 ('GM', u'Grand Master'),

 ('M', u'Master'),

 ('A', u'A'),

 ('B', u'B'),

 ('C', u'C'),

 ('D', u'D'))

IPSC\_SCORING = (('cs', u'Comstock'),

 ('vc', u'Virginia count'),

 ('ft', u'Fixed time'))

IPSC\_COURSE = (('sh', u'Short'),

 ('md', u'Medium'),

 ('lg', u'Long'),

 ('st', u'Standard'))

IPSC\_DQ\_REASON = (('no', u'Not disqualified'),

 ('gr', u'General regulations'),

 ('ad', u'Accidental discharge'),

 ('ug', u'Unsafe gun handling'),

 ('uc', u'Unsportsmanlike conduct'),

 ('pc', u'Prohibited substances'))

# Normal means a match with single firearm, tournament means a single match using two or more firearms

IPSC\_MATCH\_RULE = (('nm', u'Normal'), ('to', u'Tournament'))

IPSC\_API\_MATCH\_FIELDS = API\_MATCH\_FIELDS + ( 'level', 'firearms',

 sub\_rule', 'tournament\_division',

 handgun\_divs', 'rifle\_divs', ‘mini\_rifle\_divs’,

 prec\_rifle\_divs', 'shotgun\_divs',

 air\_divs', )

IPSC\_API\_MATCH\_EXTRAS = API\_MATCH\_EXTRAS + ('cat\_result\_limit',

 minimum\_rounds',

 maximum\_rounds',

 points',

 number\_of\_mainmatch\_competitors\_approved',

 number\_of\_mainmatch\_competitors\_pending',

 number\_of\_mainmatch\_competitors\_waiting',

 number\_of\_prematch\_competitors\_approved',

 number\_of\_prematch\_competitors\_pending',

 number\_of\_prematch\_competitors\_waiting',)

IPSC\_API\_MATCH\_EXCLUDES = API\_MATCH\_EXCLUDES

level = models.CharField(\_(u'Level'), max\_length=2, choices=ALL\_MATCH\_LEVELS, default='l1')

firearms = MultiSelectField(\_(u'Firearms'), max\_length=200, choices=ALL\_FIREARMS)

sub\_rule = models.CharField(\_(u'Sub rule'), max\_length=2, choices=IPSC\_MATCH\_RULE, default='nm')

tournament\_divisions = MultiSelectField(\_(u'Tournament divisions'), max\_length=200, choices=ALL\_SERIE\_DIVISIONS)

handgun\_divs = MultiSelectField(\_(u'Handgun divisions'), max\_length=200, choices=ALL\_FIREARM\_DIVISIONS)

rifle\_divs = MultiSelectField(\_(u'Rifle divisions'), max\_length=200, choices=ALL\_FIREARM\_DIVISIONS)

mini\_rifle\_divs = MultiSelectField(\_(u'Mini Rifle divisions'), max\_length=200, choices=ALL\_FIREARM\_DIVISIONS)

prec\_rifle\_divs = MultiSelectField(\_(u'Precision Rifle divisions'), max\_length=200, choices=ALL\_FIREARM\_DIVISIONS)

shotgun\_divs = MultiSelectField(\_(u'Shotgun divisions'), max\_length=200, choices=ALL\_FIREARM\_DIVISIONS)

air\_divs = MultiSelectField(\_(u'Air divisions'), max\_length=200, choices=ALL\_FIREARM\_DIVISIONS)

cat\_result\_limit = models.PositiveIntegerField(verbose\_name=\_(u'Limit for category results'), default=5)

### Ipsc Squad 🡨 squad

IPSC\_API\_SQUAD\_FIELDS = API\_SQUAD\_FIELDS

IPSC\_API\_SQUAD\_EXTRAS = API\_SQUAD\_EXTRAS

IPSC\_API\_SQUAD\_EXCLUDES = API\_SQUAD\_EXCLUDES

### Ipsc Competitor 🡨 competitor

IPSC\_API\_COMPETITOR\_FIELDS = API\_COMPETITOR\_FIELDS + ( 'event', 'squad',

 handgun\_div', 'handgun\_pf',

 rifle\_div', 'rifle\_pf', ‘mini\_rifle\_div’

 prec\_rifle\_div', 'prec\_rifle\_pf',

 shotgun\_div', 'air\_div',

 tournament\_division',

 category',

 classification',

 ics\_alias',

 uspsa\_num',)

IPSC\_API\_COMPETITOR\_EXTRAS = API\_COMPETITOR\_EXTRAS

IPSC\_API\_COMPETITOR\_EXCLUDES = API\_COMPETITOR\_EXCLUDES

class IpscCompetitor(Competitor):

event = models.ForeignKey(IpscMatch, verbose\_name=\_(u'Match'))

squad = models.ForeignKey(IpscSquad, verbose\_name=\_(u'Squad'), blank=True, null=True)

handgun\_div = models.CharField(\_(u'Handgun division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

handgun\_pf = models.CharField(\_(u'Handgun powerfactor'), max\_length=1, choices=IPSC\_POWERFACTOR, default='-')

rifle\_div = models.CharField(\_(u'Rifle division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

rifle\_pf = models.CharField(\_(u'Rifle powerfactor'), max\_length=1, choices=IPSC\_POWERFACTOR, default='-')

mini\_rifle\_div = models.CharField(\_(u'Min Rifle division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

prec\_rifle\_div = models.CharField(\_(u'Precision Rifle division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

prec\_rifle\_pf = models.CharField(\_(u'Precision Rifle powerfactor'), max\_length=1, choices=IPSC\_POWERFACTOR, default='-')

shotgun\_div = models.CharField(\_(u'Shotgun division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

air\_div = models.CharField(\_(u'Air division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

tournament\_division = models.CharField(\_(u'Tournament division'), max\_length=5, choices=ALL\_SERIE\_DIVISIONS, blank=True)

category = models.CharField(\_(u'Category'), max\_length=2, choices=ALL\_CATEGORIES, default='-')

classification = models.CharField(\_(u'Classification'), max\_length=2, choices=IPSC\_USPSA\_CLASSIFICATION, default='U')

 # ICS Alias is a worldwide ID within IPSC

 # The ICS alias is not case sensitive (=stored as lower case)

 # It must be from 4 to 16 alphanumeric characters (A-Z and 0-9 only, no spaces or special characters).

ics\_alias = models.CharField(\_(u'ICS Alias'), max\_length=16, blank=True)

 # uspsa num is as ICS Alias but for USPSA, see user profile for full details

uspsa\_num = models.CharField(\_(u'USPSA number'), max\_length=10, blank=True)

### Ipsc Stage 🡨 stage

IPSC\_API\_STAGE\_FIELDS = API\_STAGE\_FIELDS + ( 'event', 'scoring',

 course', 'paper',

 bonus', 'popper',

 plate', 'frangible',

 penalty',

 minimum\_rounds',

 maximum\_rounds',

 standard',

 firearm', )

IPSC\_API\_STAGE\_EXTRAS = API\_STAGE\_EXTRAS + ('max\_points', 'match\_percent',)

IPSC\_API\_STAGE\_EXCLUDES = API\_STAGE\_EXCLUDES

event = models.ForeignKey(IpscMatch, verbose\_name=\_(u'Match'))

scoring = models.CharField(\_(u'Scoring'), max\_length=2, choices=IPSC\_SCORING, default='cs')

course = models.CharField(\_(u'Course'), max\_length=2, choices=IPSC\_COURSE, default='sh')

paper = models.PositiveIntegerField(\_('Paper targets'), default=0)

bonus = models.PositiveIntegerField(\_('Bonus Paper targets'), default=0)

popper = models.PositiveIntegerField(\_('Poppers'), default=0)

plate = models.PositiveIntegerField(\_('Plates'), default=0)

frangible = models.PositiveIntegerField(\_('Frangibles'), default=0)

penalty = models.PositiveIntegerField(\_('No-Shoots'), default=0)

minimum\_rounds = models.PositiveIntegerField(\_('Minimum Rounds'), default=0) # rounds needed for all except disapperaing/bonus targets

maximum\_rounds = models.PositiveIntegerField(\_('Maximum Rounds'), default=0) # same as min except when using disappering targets

standard = models.ForeignKey(IpscStandardStage, verbose\_name=\_(u'Classifier stage'), related\_name='%(class)s\_standard\_set', blank=True, null=True, default=None)

firearm = models.CharField(\_(u'Firearm'), max\_length=2, choices=ALL\_FIREARMS, default='hg')

### ipsc Scorecard (aka Score Sheet) 🡨 scorecard

IPSC\_API\_SCORECARD\_FIELDS = API\_SCORECARD\_FIELDS + ( 'stage',

 competitor',

 ascore',

 bscore',

 cscore',

 dscore',

 miss',

 penalty',

 procedural',

 time',

 zeroed',

 spec\_penalty',

 score',

 deductions',

 points',

 hitfactor',

 dq\_reason',

 comment',

 ‘warning’,
 ‘incomplete’,)

IPSC\_API\_SCORECARD\_EXTRAS = API\_SCORECARD\_EXTRAS

IPSC\_API\_SCORECARD\_EXCLUDES = API\_SCORECARD\_EXCLUDES + ('comment',)

stage = models.ForeignKey(IpscStage, verbose\_name=\_(u'Stage'))

competitor = models.ForeignKey(IpscCompetitor, verbose\_name=\_(u'Competitor'))

ascore = models.PositiveIntegerField(\_(u'A hits'), default=0)

bscore = models.PositiveIntegerField(\_(u'B hits'), default=0)

cscore = models.PositiveIntegerField(\_(u'C hits'), default=0)

dscore = models.PositiveIntegerField(\_(u'D hits'), default=0)

miss = models.PositiveIntegerField(\_(u'Miss (-10)'), default=0)

penalty = models.PositiveIntegerField(\_(u'NS (-10)'), default=0)

procedural = models.PositiveIntegerField(\_(u'Proc (-10)'), default=0)

time = models.DecimalField(verbose\_name=\_(u'Time (sec)'), max\_digits=5, decimal\_places=2, default=0)

tnh = models.PositiveIntegerField(\_(u'Target not hit (TNH -5)'), default=0) # time-plus only

tnn = models.PositiveIntegerField(\_(u'Target not neutralized (TNN -10)'), default=0) # time-plus only

tne = models.PositiveIntegerField(\_(u'Target not engaged (TNE -15)'), default=0) # time-plus only

evt = models.CommaSeparatedIntegerField(\_(u'Enhanced Value Targets hits (10-40p)'), max\_length=50, default=EVT\_NO\_ENTRIES) # time-plus only

stage\_not\_fired = models.BooleanField(\_(u'Stage not fired (SNF 500 sec)'), default=False) # time-plus only

timeplus = models.DecimalField(verbose\_name=\_(u'Time Plus (sec)'), max\_digits=5, decimal\_places=2, default=0) # sum of all time + bonus time of hitfactor

pf\_correction = models.PositiveIntegerField(\_(u'PF correction (San Angelo)'), default=0) # used in multigun where stage has different firearms and pf

zeroed = models.BooleanField(\_(u'Zeroed stage'), default=False)

incomplete = models.BooleanField(\_(u'Incomplete hits/misses (9.7.6)'), default=False) # rule 9.7.6.2 insufficient hits or misses recorded on the score sheet

warning = models.BooleanField(\_(u'Received warning'), default=False)

 spec\_penalty = models.PositiveIntegerField(\_(u'Special penalty (1-20%)'), default=0)

score = models.PositiveIntegerField(\_(u'Score (a-d)'), default=0) # the total score from all A-D

deductions = models.PositiveIntegerField(\_(u'Deductions'), default=0)

points = models.PositiveIntegerField(\_(u'Points'), default=0) # points = score - deductions

hitfactor = models.DecimalField(\_(u'Hit factor'), max\_digits=12, decimal\_places=7, blank=True, null=True, default=0)

dq\_reason = models.CharField(\_(u'DQ reason'), max\_length=2, choices=IPSC\_DQ\_REASON, default='no')

comment = models.CharField(\_(u'Comment'), max\_length=100, blank=True)

### Ipsc Team 🡨 team

TEAM\_ALL\_CATEGORIES = ((('---'), \_(u'not specified, allow any category')),) + ALL\_CATEGORIES

TEAM\_IPSC\_CATEGORY = ((('---'), \_(u'not specified, allow any category')),) + IPSC\_CATEGORY

TEAM\_USPSA\_CATEGORY = ((('---'), \_(u'not specified, allow any category')),) + USPSA\_CATEGORY

TEAM\_IPSC\_USPSA\_CLASSIFICATION = ((('---'), \_(u'not specified, allow any classification')),) + IPSC\_USPSA\_CLASSIFICATION

IPSC\_API\_TEAM\_FIELDS = API\_TEAM\_FIELDS + ( 'event',

 handgun\_div',

 rifle\_div', ‘mini\_rifle\_div’,

 prec\_rifle\_div',

 shotgun\_div',

 tournament\_div',

 category',

 classification',)

IPSC\_API\_TEAM\_EXTRAS = API\_TEAM\_EXTRAS

IPSC\_API\_TEAM\_EXCLUDES = API\_TEAM\_EXCLUDES

event = models.ForeignKey(IpscMatch, verbose\_name=\_(u'Match'))

handgun\_div = models.CharField(\_(u'Handgun division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

rifle\_div = models.CharField(\_(u'Rifle division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

mini\_rifle\_div = models.CharField(\_(u'Mini Rifle division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

prec\_rifle\_div = models.CharField(\_(u'Precision Rifle division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

shotgun\_div = models.CharField(\_(u'Shotgun division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

air\_div = models.CharField(\_(u'Air division'), max\_length=5, choices=ALL\_FIREARM\_DIVISIONS, blank=True)

tournament\_division = models.CharField(\_(u'Tournament division'), max\_length=5, choices=ALL\_SERIE\_DIVISIONS, blank=True)

category = models.CharField(\_(u'Team Category'), max\_length=3, choices=TEAM\_ALL\_CATEGORIES, default='---')

classification = models.CharField(\_(u'Team Classification'), max\_length=3, choices=TEAM\_IPSC\_USPSA\_CLASSIFICATION, default='---')

### IPSC Team Member 🡨 team member

IPSC\_API\_TEAMMEMBER\_FIELDS = API\_TEAMMEMBER\_FIELDS

IPSC\_API\_TEAMMEMBER\_EXTRAS = API\_TEAMMEMBER\_EXTRAS

IPSC\_API\_TEAMMEMBER\_EXCLUDES = API\_TEAMMEMBER\_EXCLUDES

team = models.ForeignKey(IpscTeam, verbose\_name=\_(u'Team'))

competitor = models.ForeignKey(IpscCompetitor, verbose\_name=\_(u'Competitor'))

# IPSC/USPSA Series

More info will be added about this.

## URLs

/ipsc/serie/<key>/ [GET]

/ipsc/serie/<key>/all/ [GET]

/ipsc/serie/competitor/<key>/ [GET]

/ipsc/serie/<key>/competitors/ [GET]

/ipsc/serie/<key>/componentmatches/ [GET]

Note <key> in all above expresses the primary key of the instance accessed.

# IDPA Matches

## URLs

/idpa/match/<key>/ [GET]

/idpa/match/<key>/all/ [GET]

/ipsc/match/<key>/stats/ [GET]

/idpa/stage/<key>/ [GET]

/idpa/competitor/<key>/ [GET]

/idpa/competitor/<key>/all/ [GET]

/idpa/competitor/<key>/set\_pin/ [POST]

/idpa/competitor/<key>/reset\_pin/ [GET]

/idpa/competitor/<key>/verify\_all/ [POST]

/idpa/scoresheet/<key>/ [GET DELETE]

/idpa/scoresheet/<key>/verify/ [POST]

/idpa/scoresheet/<key>/unverify/ [GET PUT POST]

/idpa/squad/<key>/ [GET]

/idpa/match/<key>/competitors/ [GET]

/idpa/match/<key>/stages/ [GET]

/idpa/match/<key>/squads/ [GET]

/idpa/match/<key>/scoresheets/ [POST GET PUT]

/idpa/stage/<key>/scoresheets/ [GET]

/idpa/competitor/<key>/scoresheets [GET]

## Creating or updating score sheets

Each POST or PUT, i.e. create or update of a score sheet must contain all the fields below. PK shall only be expressed in the case you are updating an existing score sheet, if creating a new ensure you provide pk for stage and competitor.

The array can contain one or more score sheets – if you post/put multiple then all will be checked and if and only if all are ok will they be updated or created.

 TBD

POST or PUT for scorecard must contain the following fields and these are the only ones that will be considered; 'competitor', 's1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 'points\_down', 'procedural', 'hnt', 'ftn', 'ftdr', 'dnf', 'dq\_reason', 'comment'.

If any field is not applicable then set to default value.

## Model, fields, extras and excluded

IDPA\_MATCH\_LEVELS = ( ('l1', u'Level I'),

 ('l2', u'Level II'),

 ('l3', u'Level III'),

 ('nt', u'Nationals'))

IDPA\_FIREARM\_HANDGUN\_DIVISIONS = ( ('SSP', u'Stock Service Pistol'),

 ('ESP', u'Enhanced Service Pistol'),

 ('CDP', u'Custom Defensive Pistol'),

 ('ESR', u'Enhanced Service Revolver'),

 ('SSR', u'Stock Service Revolver'),

 ('BUG', u'Back-Up Gun'))

IDPA\_FIREARM\_DMG\_DIVISIONS = ( ('SSG', u'Stock Service Gun'),

 ('ESG', u'Enhanced Service Gun'),

 ('SPCC', u'Stock Pistol Caliber Carbine'),

 ('EPCC', u'Enhanced Pistol Caliber Carbine'))

IDPA\_FIREARMS = ( ('hg', u'Handgun'),

 ('sg', u'Shootgun'),

 ('rf', u'Rifle'))

IDPA\_CATEGORY = ( ('-', u'None'),

 ('L', u'Lady'),

 ('J', u'Junior'),

 ('S', u'Senior'),

 ('SS', u'Super Senior'),

 ('I', u'Industry'),

 ('P', u'Press'),

 ('MI', u'Military (active)'),

 ('MIR', u'Military (retired)'),

 ('LE', u'Law Enforcement (active)'),

 ('LER', u'Law Enforcement (retired)'))

IDPA\_CLASSIFICATION = ( ('UN', u'Unclassified'),

 ('NV', u'Novice'),

 ('MM', u'Marksman'),

 ('SS', u'Sharpshooter'),

 ('EX', u'Expert'),

 ('MA', u'Master'),

 ('DM', u'Distingushed Master'))

IDPA\_SCORING = ( ('vc', u'Vickers Count'),

 ('lc', u'Limited Vickers Count'))

IDPA\_DQ\_REASON = ( ('no', u'Not disqualified'),

 ('ug', u'Unsafe gun handling'),

 ('uc', u'Unsportsmanlike conduct'),

 ('am', u'Ammunition fails to meet minmum power floor'),

 ('gr', u'General regulations'),

 ('ad', u'Accidental discharge'),

 ('pc', u'Prohibited substances'))

# Normal means a match with single firearm, tournament means a single match using two or more firearms

IDPA\_MATCH\_RULE = (('hg', u'Handgun'), ('dmg', u'Defensive Multi Gun'))

### idpa Match

IDPA\_API\_MATCH\_FIELDS = API\_MATCH\_FIELDS + ('level',

 sub\_rule',

 handgun\_divs',

 dmg\_divs',

 category' )

IDPA\_API\_MATCH\_EXTRAS = API\_MATCH\_EXTRAS + ( 'minimum\_rounds', )

IDPA\_API\_MATCH\_EXCLUDES = API\_MATCH\_EXCLUDES

level = models.CharField(\_(u'Level'), max\_length=2, choices=IDPA\_MATCH\_LEVELS, default='l1')

sub\_rule = models.CharField(\_(u'Sub rule'), max\_length=3, choices=IDPA\_MATCH\_RULE, default='hg')

handgun\_divs = MultiSelectField(\_(u'Recognized divisions'), max\_length=200, choices=IDPA\_FIREARM\_HANDGUN\_DIVISIONS, default='SSP')

dmg\_divs = MultiSelectField(\_(u'Recognized divisions'), max\_length=200, choices=IDPA\_FIREARM\_DMG\_DIVISIONS, default='SSG')

category = MultiSelectField(\_(u'Recognized Categories'), max\_length=200, choices=IDPA\_CATEGORY, default='-')

### idpa Competitor

IDPA\_API\_COMPETITOR\_FIELDS = API\_COMPETITOR\_FIELDS + ( 'squad',

 handgun\_div',

 dmg\_div',

 category',

 classification',

 idpa\_num',)

IDPA\_API\_COMPETITOR\_EXTRAS = API\_COMPETITOR\_EXTRAS

IDPA\_API\_COMPETITOR\_EXCLUDES = API\_COMPETITOR\_EXCLUDES

event = models.ForeignKey(IdpaMatch, verbose\_name=\_(u'Match'))

squad = models.ForeignKey(IdpaSquad, verbose\_name=\_(u'Squad'), blank=True, null=True)

handgun\_div = models.CharField(\_(u'Handgun division'), max\_length=5, choices=IDPA\_FIREARM\_HANDGUN\_DIVISIONS, blank=True)

dmg\_div = models.CharField(\_(u'Rifle division'), max\_length=5, choices=IDPA\_FIREARM\_DMG\_DIVISIONS, blank=True)

category = models.CharField(\_(u'Category'), max\_length=2, choices=IDPA\_CATEGORY, default='-')

classification = models.CharField(\_(u'Classification'), max\_length=2, choices=IDPA\_CLASSIFICATION, default='UN')

# Please enter your IDPA # as it appears on your IDPA Membership card.

# The format should be either "A00000" or "F00000".

# A - american, F - Foreigner

idpa\_num = models.CharField(\_(u'IDPA num'), max\_length=6, blank=True)

### idpa Stage

IDPA\_API\_STAGE\_FIELDS = API\_STAGE\_FIELDS + ( 'event',

 scoring',

 firearms',

 paper',

 popper',

 plate',

 frangible',

 penalty',

 strings',

 minimum\_rounds',

 standard',)

IDPA\_API\_STAGE\_EXTRAS = API\_STAGE\_EXTRAS

IDPA\_API\_STAGE\_EXCLUDES = API\_STAGE\_EXCLUDES

event = models.ForeignKey(IdpaMatch, verbose\_name=\_(u'Match'))

scoring = models.CharField(\_(u'Scoring'), max\_length=2, choices=IDPA\_SCORING, default='vc') # Vickers Count most used.

firearms = MultiSelectField(\_(u'Fireams'), max\_length=50, choices=IDPA\_FIREARMS, default='hg')

paper = models.PositiveIntegerField(\_('Paper targets'), default=0)

popper = models.PositiveIntegerField(\_('Poppers'), default=0)

plate = models.PositiveIntegerField(\_('Plates'), default=0)

frangible = models.PositiveIntegerField(\_('Frangibles'), default=0)

penalty = models.PositiveIntegerField(\_('No-Shoots'), default=0)

strings = models.PositiveIntegerField(\_('Strings'), default=1) # strings can be in range 1...10

minimum\_rounds = models.PositiveIntegerField(\_('Minimum Rounds'), default=0) # minimum is sort of, it is really the exact amount of round that shall be scored.

standard = models.ForeignKey(IdpaStandardStage, verbose\_name=\_(u'Classifier stage'), related\_name='%(class)s\_standard\_set', blank=True, null=True, default=None)

### idpa Scoresheet

IDPA\_DNF = (('x', u'scored stage'), ('c1', u'Rule C.1'), ('c2', u'Rule C.2'))

IDPA\_API\_SCORECARD\_FIELDS = API\_SCORECARD\_FIELDS + ( 'stage',
 'competitor',
 'points\_down',
 'procedural',

 hnt',

 ftn',

 ftdr',

 score',

 s1','s2','s3','s4','s5','s6'

 s7','s8','s9','s10','s11','s12'

 dnf',

 dq\_reason',)

IDPA\_API\_SCORECARD\_EXTRAS = API\_SCORECARD\_EXTRAS + ('string\_times', raw\_time',

 incurred\_time',)

IDPA\_API\_SCORECARD\_EXCLUDES = API\_SCORECARD\_EXCLUDES + ('comment',)

stage = models.ForeignKey(IdpaStage, verbose\_name=\_(u'Stage'))
competitor = models.ForeignKey(IdpaCompetitor, verbose\_name=\_(u'Competitor'))
points\_down = models.PositiveIntegerField(\_(u'Points down (x 0.5 sec)'), default=0) procedural = models.PositiveIntegerField(\_(u'Procedural (x 3.0 sec)'), default=0)
hnt = models.PositiveIntegerField(\_(u'Hit on non-threat target (x 5.0 sec)'), default=0)
ftn = models.PositiveIntegerField(\_(u'Failure to neutralize (x 5.0 sec)'), default=0)
ftdr = models.PositiveIntegerField(\_(u'Failure to do right (x 20.0 sec)'), default=0)
score = models.DecimalField(verbose\_name=\_(u'Score'), max\_digits=6, decimal\_places=2, default=0)

s1 = models.DecimalField(verbose\_name=\_(u'String 1 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s2 = models.DecimalField(verbose\_name=\_(u'String 2 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s3 = models.DecimalField(verbose\_name=\_(u'String 3 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s4 = models.DecimalField(verbose\_name=\_(u'String 4 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s5 = models.DecimalField(verbose\_name=\_(u'String 5 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s6 = models.DecimalField(verbose\_name=\_(u'String 6 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s7 = models.DecimalField(verbose\_name=\_(u'String 7 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s8 = models.DecimalField(verbose\_name=\_(u'String 8 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s9 = models.DecimalField(verbose\_name=\_(u'String 9 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s10 = models.DecimalField(verbose\_name=\_(u'String 10 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s11 = models.DecimalField(verbose\_name=\_(u'String 11 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)
s12 = models.DecimalField(verbose\_name=\_(u'String 12 Time (sec)'), max\_digits=5, decimal\_places=2, default=0)

dnf = models.BooleanField(\_(u'DNF according to App 4 Sect C'), default=False)
dq\_reason = models.CharField(\_(u'DQ reason'), max\_length=2, choices=IDPA\_DQ\_REASON, default='no')
comment = models.CharField(\_(u'Comment'), max\_length=100, blank=True)

### IDPA SQUAD

IDPA\_API\_SQUAD\_FIELDS = API\_SQUAD\_FIELDS
IDPA\_API\_SQUAD\_EXTRAS = API\_SQUAD\_EXTRAS
IDPA\_API\_SQUAD\_EXCLUDES = API\_SQUAD\_EXCLUDES

event = models.ForeignKey(IdpaMatch, verbose\_name=\_(u'Match'))

### IDPA Team and Team member

Not applicable

# PPC Matches

## URLs

/ppc/match/<key>/ [GET] – get info about a mach

/ppc/match/<key>/all/ [GET] – get info about a mach

/ipsc/match/<key>/stats/ [GET]

/ppc/competitor/<key>/ [GET] – get all info about a competitor incl. score

/ppc/competitor/<key>/clear/ [DELETE] – clear/reset all scores for competitor

/ppc/competitor/<key>/set\_pin/ [POST]

/ppc/competitor/<key>/reset\_pin/ [GET]

/ppc/competitor/<key>/verify/ [GET POST PUT]

/ppc/competitor/<key>/unverify/ [GET]

/ppc/squad/<key>/ [GET] – get all info about a squad

/ppc/match/<key>/competitors/ [GET] – get all competitors in match

/ppc/match/<key>/squads/ [GET] – get all squads

/ppc/match/<key>/scoresheets/ [POST GET PUT] – get all scoresheets or create/update them

Note that the in PPC the competitor model has both competitor information and the score of this competitor but they are updated or received using different URL since there are different RBAC rules regarding editing a competitor vs. scoring a competitor.

## Creating or updating score sheets

Each POST or PUT, i.e. create or update of a score sheet must contain all the fields below. PK shall only be expressed in the case you are updating an existing score sheet, if creating a new – then no pk. The array can contain one or more score sheets – if you post/put multiple then all will be checked and iff all are ok will they be updated or created.

Since competitor information (name etc) and score is stored in same object you still need to post to different URL depending on what you intend to do.

 [

 {

 "pk": 1,

 "model": "match\_ppc.ppc\_competitor",

 "s1": "30,3,3,3,3,0,0",

 "s2": "30,3,3,3,3,0,0",

 "s3": "30,3,3,3,3,0,0",

 "s5": "30,3,3,3,3,0,0",

 "s5": "30,3,3,3,3,0,0",

 "s6": "30,3,3,3,3,0,0",

 "dq\_reason": "no",

 "score\_comment": "",

 },

 {

 "pk": 2,

 "model": "match\_ppc.ppc\_competitor",

 "s1": "30,3,3,3,3,0,0",

 "s2": "30,3,3,3,3,0,0",

 "s3": "30,3,3,3,3,0,0",

 "s5": "30,3,3,3,3,0,0",

 "s5": "30,3,3,3,3,0,0",

 "s6": "30,3,3,3,3,0,0",

 "dq\_reason": "no",

 "score\_comment": "",

 }

]

For details on field formats, allowed lengths, DQ codes etc. check reference on Competitor below on. (note – score sheets is the term used in GUI but internally for historical reasons the model and classes are called scorecards).

## Model, fields, extras and excluded

PPC\_MATCH\_TYPE = ( ('wa', u'WA1500'), ('nr', u'NRA'))

PPC\_WEAPON\_CLASS = [ ('mm', u'Revolver 1500 (R1500)'),

 ('ap', u'Pistol 1500 (P1500)'),

 ('dy', u'Duty 1500 (D1500)'),

 ('o2', u'Open 1200 25 yrds (O1225)'),

 ('o5', u'Open 1200 50 yrds (O1250)'),

 ('dr', u'Distinguished Revolver (DR)'),

 ('dp', u'Distinguished Pistol (DP)'),

 ('ss', u'Stock Semi-Auto Pistol (SSA)'),

 ('op', u'Open (OP)'),

 ('sr', u'Service Revolver (SR)'),

 ('od', u'Off Duty Revolver (ODR)'),

 ('sd', u'Off Duty Revolver 5-shot (ODR5)'),

 ('sn', u'Snub Nose (SN)') ]

PPC\_CLASSIFICATION = (('5', u'High Master'),

 ('4', u'Master'),

 ('3', u'Expert'),

 ('2', u'Sharpshooter'),

 ('1', u'Marksman'),

 ('6', u'Temporary'),

 ('0', u'Unclassified'))

PPC\_DQ\_REASON = ( ('no', u'Not disqualified'),

 ('fu', u'Pointing firearm uprange or at another'),

 ('dr', u'Dropping a loaded firearm'),

 ('du', u'Dropping a unloaded firearm'),

 ('ll', u'Leaving the line with loaded firearm'),

 ('ad', u'Accidental discharge'),

 ('pu', u'Other unsafe or incompetent gun handling'),

 ('sh', u'Shot fired in holster, barricade, air or in ground'),

 ('ug', u'Unsafe gun handling'),

 ('dd', u'Under influence of controlled substance (drug or alcohol)'),

 ('uc', u'Unsportsmanlike conduct'),

 ('pc', u'Disrespectful words and/or gestures'))

PPC\_API\_MATCH\_FIELDS = API\_MATCH\_FIELDS + ( 'match\_type',

 weapon\_classes',)

PPC\_API\_MATCH\_EXTRAS = API\_MATCH\_EXTRAS + ( 'registration\_url',)

PPC\_API\_MATCH\_EXCLUDES = API\_MATCH\_EXCLUDES

### Ppc Match

match\_type = models.CharField(\_(u'Match'), max\_length=2, choices=PPC\_MATCH\_TYPE, default='wa')

weapon\_classes = MultiSelectField(\_(u'Weapons classes'), max\_length=100, choices=PPC\_WEAPON\_CLASS)

# ---------------------------------------------------------------------------

PPC\_API\_SQUAD\_FIELDS = API\_SQUAD\_FIELDS

PPC\_API\_SQUAD\_EXTRAS = API\_SQUAD\_EXTRAS

PPC\_API\_SQUAD\_EXCLUDES = API\_SQUAD\_EXCLUDES

### Ppc Squad

event = models.ForeignKey(PpcMatch, verbose\_name=\_(u'Match'))

weapon\_class = models.CharField(\_(u'Weapon class'), max\_length=2, choices=PPC\_WEAPON\_CLASS, blank=True, default='')

###

### PPC COMPETITOR

One difference between other sports and PPC is that in PPC it is always fixed what target/rounds you shoot depending on your weapon class. This means we store the scores in the competitor and not in scorecard as there is no need for stage or scorecards.

The competitor shoots and scores in the strings or s1 – s6 comma separated integer fields where each position in the string is number of hits of some point (X, 10, … and so on).

The type number of strings and shot depends on weapon class and are as follows (just some code cut out to show the logic, the initial string like M1, T1 and so on is the name for string;

if self.is\_150\_shot\_course:
 # max hits per string [24,18,24,24,30,30]
 strings.append(['M1', self.target1, self.t1\_points]) strings.append(['M2', self.target2, self.t2\_points])
 strings.append(['M3', self.target3, self.t3\_points])
 strings.append(['M4', self.target4, self.t4\_points])
 strings.append(['S 1-2', self.target5, self.t5\_points])
 strings.append(['S 3-4', self.target6, self.t6\_points])

elif self.is\_120\_25\_yrd\_course or self.is\_120\_50\_yrd\_course:
 # max hits per string [36,24,36,24,0,0] or [30,30,30,30,0,0]
 strings.append(['T1', self.target1, self.t1\_points])
 strings.append(['T2', self.target2, self.t2\_points])
 strings.append(['T3', self.target3, self.t3\_points])
 strings.append(['T4', self.target4, self.t4\_points])

elif self.is\_shotgun\_course:

 # max hits (shotgun – with buckshot) [45,45,5,5]
 strings.append(['S1', self.target1, self.t1\_points])
 strings.append(['S2', self.target2, self.t2\_points])
 strings.append(['S3', self.target3, self.t3\_points])
 strings.append(['S4', self.target4, self.t4\_points])
else:

# 60 shots [30,30,0,0,0,0] if weapon class in['dp', 'dr', 'op', 'rp', 'sn']
# else 48 shots [18,30,0,0,0,0] if weapon class in ['ss', 'sr', 'od', 'ox', 'sd']
 strings.append(['S 1-2', self.target1, self.t1\_points])
 strings.append(['S 3-4', self.target2, self.t2\_points])

--

PPC\_API\_SCORECARD\_FIELDS = ('score\_comment',

 weapon\_class',

 dq\_reason',

 s1',

 s2',

 s3',

 s4',

 s5',

 s6',)

PPC\_API\_SCORECARD\_EXTRAS = API\_SCORECARD\_EXTRAS + ( 'max\_hits\_targets',

 is\_scoring\_complete',

 ‘total\_ten’,

 ‘total\_x’,

 ‘verified’,

 ‘is\_verified')

PPC\_API\_SCORECARD\_EXCLUDES = API\_SCORECARD\_EXCLUDES + ('comment',

 ‘score\_comment)

PPC\_API\_COMPETITOR\_FIELDS = API\_COMPETITOR\_FIELDS + ( 'event',

 squad',

 weapon\_class',

 classification',

 waid',)

PPC\_API\_COMPETITOR\_EXTRAS = API\_COMPETITOR\_EXTRAS

PPC\_API\_COMPETITOR\_EXCLUDES = API\_COMPETITOR\_EXCLUDES

PPC\_EMPTY\_STRING = '0,0,0,0,0,0,0'

event = models.ForeignKey(PpcMatch, verbose\_name=\_(u'Match'))

squad = models.ForeignKey(PpcSquad, verbose\_name=\_(u'Squad'), blank=True, null=True)

weapon\_class = models.CharField(\_(u'Class'), max\_length=2, choices=PPC\_WEAPON\_CLASS, blank=True)

classification = models.CharField(\_(u'Classification'), max\_length=2, choices=PPC\_CLASSIFICATION, default='0')

# World Association PPC 1500 ID

# is just a regular number - unknown standard but seems to be only digits and max 5 but no standard exists

waid = models.CharField(\_(u'WA1500 Id'), max\_length=8, blank=True)

# each string is xxx, ten, nine, eight, seven, zero, miss (7 values)

s1 = models.CommaSeparatedIntegerField(\_(u'String 1'), max\_length=50, default=PPC\_EMPTY\_STRING)

s2 = models.CommaSeparatedIntegerField(\_(u'String 2'), max\_length=50, default=PPC\_EMPTY\_STRING)

s3 = models.CommaSeparatedIntegerField(\_(u'String 3'), max\_length=50, default=PPC\_EMPTY\_STRING)

s4 = models.CommaSeparatedIntegerField(\_(u'String 4'), max\_length=50, default=PPC\_EMPTY\_STRING)

s5 = models.CommaSeparatedIntegerField(\_(u'String 5'), max\_length=50, default=PPC\_EMPTY\_STRING)

s6 = models.CommaSeparatedIntegerField(\_(u'String 6'), max\_length=50, default=PPC\_EMPTY\_STRING)

dq\_reason = models.CharField(\_(u'DQ reason'), max\_length=2, choices=PPC\_DQ\_REASON, default='no')

score\_comment = models.CharField(\_(u'Scoring comment'), max\_length=300, blank=True)

### PPC Team and Team Member

Not applicable

# PPC Series

More info will be added here.

## URLs

/ppc/serie/<key>/ [GET]

/ppc/serie/<key>/all/ [GET]

/ppc/serie/competitor/<key>/ [GET]

/ppc/serie/<key>/competitors/ [GET]

/ppc/serie/<key>/componentmatches/ [GET]

Note <key> in all above expresses the primary key of the instance accessed.

# Steel Matches

## URLs

/steel/match/<key>/ [GET]

/steel/match/<key>/all/ [GET]

/ipsc/match/<key>/stats/ [GET]

/steel/stage/<key>/ [GET]

/steel/competitor/<key>/ [GET]

/steel/competitor/<key>/all/ [GET]

/steel/competitor/<key>/set\_pin/ [POST]

/steel/competitor/<key>/reset\_pin/ [GET]

/steel/competitor/<key>/verify\_all/ [POST]

/steel/scoresheet/<key>/ [GET DELETE]

/steel/scoresheet/<key>/verify/ [POST]

/steel/scoresheet/<key>/unverify/ [GET PUT POST]

/steel/squad/<key>/ [GET]

/steel/match/<key>/competitors/ [GET]

/steel/match/<key>/stages/ [GET]

/steel/match/<key>/squads/ [GET]

/steel/match/<key>/scoresheets/ [POST GET PUT]

/steel/stage/<key>/scoresheets/ [GET]

/steel/competitor/<key>/scoresheets [GET]

## Creating or updating score sheets

**To be defined**

Currently String is a separate entity from the score sheet, but strings will be merged into the scoresheet to form one entity.

## Model, fields, extras and excluded

STEEL\_FIREARM\_DIVISION = ( ('prd', u'Production'),

 ('irp', u'Iron Sights'),

 ('opp', u'Open'),

 ('opr', u'Optic Revolver'),

 ('isr', u'Iron Sight Revolver'),

 ('rio', u'Rimfire Open'),

 ('ris', u'Rimfire Iron Sights'),

 ('cas', u'Cowboy Single Action'),

 ('ist', u'IPSC Standard'),

 ('usl', u'USPSA Limited'),

 ('icd', u'IDPA Custom Defensive Pistol'),

 ('isc', u'IDPA Stock Service Pistol'),

 ('ies', u'IDPA Enhanced Service Pistol')

 )

 # divisions marked with \* are not according to SCSA Rulebook 2011

STEEL\_CATEGORY = (

 ('S', u'Senior'),

 ('SS', u'Super Senior'),

 ('L', u'Lady'),

 ('LE', u'Law Enforcement'),

 ('M', u'Military'),

 ('PJ', u'Pre-Teen'),

 ('J', u'Junior')

 )

STEEL\_SCORING = ( ('s', u'sound'),

 ('i', u'impact'),

 ('c', u'sound w. correction'))

STEEL\_STD\_SCORING = ( ('c', u'sound w. correction'),

 ('i', u'impact'))

STEEL\_LEVEL = ( ('T1', u'Tier-1 (Local)'),

 ('T2', u'Tier-2 (State)'),

 ('T3', u'Tier-3 (Regional)'),

 ('T4', u'Tier-4 (World Championships)'))

STEEL\_DQ\_REASON = ( ('no', u'Not disqualified'),

 ('fu', u'Pointing firearm uprange (breaking 180)'),

 ('dr', u'Dropping a loaded firearm'),

 ('ll', u'Leaving the line with loaded firearm'),

 ('ad', u'Accidental discharge'),

 ('pu', u'Picking up a dropped firearm without direction of match official'),

 ('sh', u'Shot fired in holster or in ground'),

 ('ug', u'Unsafe gun handling'),

 ('dd', u'Under influence of controlled substance (drug or alcohol)'),

 ('uc', u'Unsportsmanlike conduct'),

 ('pc', u'Disrespectful words and/or gestures'))

### Steel Match

STEEL\_API\_MATCH\_FIELDS = API\_MATCH\_FIELDS + ('level', 'divisions', )

STEEL\_API\_MATCH\_EXTRAS = API\_MATCH\_EXTRAS + ( 'minimum\_rounds',)

STEEL\_API\_MATCH\_EXCLUDES = API\_MATCH\_EXCLUDES

divisions = MultiSelectField(\_(u'Divisions'), max\_length=150, choices=STEEL\_FIREARM\_DIVISION)

level = models.CharField(\_(u'Level'), max\_length=2, choices=STEEL\_LEVEL, default='T1')

### Steel Competitor

STEEL\_API\_COMPETITOR\_FIELDS = API\_COMPETITOR\_FIELDS + ( 'event', 'squad',

 division',

 categories',)

STEEL\_API\_COMPETITOR\_EXTRAS = API\_COMPETITOR\_EXTRAS

STEEL\_API\_COMPETITOR\_EXCLUDES = API\_COMPETITOR\_EXCLUDES

event = models.ForeignKey(SteelMatch, verbose\_name=\_(u'Match'))

squad = models.ForeignKey(SteelSquad, verbose\_name=\_(u'Squad'), blank=True, null=True)

division = models.CharField(\_(u'Division'), max\_length=3, choices=STEEL\_FIREARM\_DIVISION, blank=True)

categories = MultiSelectField(\_(u'Category'), blank=True, max\_length=150, choices=STEEL\_CATEGORY)

### Steel Stage

STEEL\_API\_STAGE\_FIELDS = API\_STAGE\_FIELDS + ( 'event', 'scoring',

 string', 'count',

 sound\_correction', 'stop\_plate',

 standard',)

STEEL\_API\_STAGE\_EXTRAS = API\_STAGE\_EXTRAS + ('get\_sound\_correction\_time', 'minimum\_rounds')

STEEL\_API\_STAGE\_EXCLUDES = API\_STAGE\_EXCLUDES

event = models.ForeignKey(SteelMatch, verbose\_name=\_(u'Match'))

scoring = models.CharField(\_(u'Scoring'), max\_length=2, choices=STEEL\_SCORING, default='s')

string = models.PositiveIntegerField(\_('Strings'), default=5)

count = models.PositiveIntegerField(\_('Counted strings'), default=4)

stop\_plate = models.PositiveIntegerField(\_('Distance to stop plate (feet)'), default=30)

sound\_correction = models.DecimalField(\_(u'Flight time factor (sec per string)'), max\_digits=3, decimal\_places=2, default=0)

standard = models.ForeignKey(SteelStandardStage, verbose\_name=\_(u'Classifier stage'), related\_name='%(class)s\_standard\_set', blank=True, null=True)

### Steel Scoresheet

STEEL\_API\_SCORECARD\_FIELDS = API\_SCORECARD\_FIELDS + ( 'stage',
 competitor',
 dq\_reason',
 comment',)

STEEL\_API\_SCORECARD\_EXTRAS = API\_SCORECARD\_EXTRAS + ('total\_time', 'average\_time')
STEEL\_API\_SCORECARD\_EXCLUDES = API\_SCORECARD\_EXCLUDES + ('comment',)

stage = models.ForeignKey(SteelStage, verbose\_name=\_(u'Stage'))
competitor = models.ForeignKey(SteelCompetitor, verbose\_name=\_(u'Competitor'))
dq\_reason = models.CharField(\_(u'DQ reason'), max\_length=2, choices=STEEL\_DQ\_REASON, default='no')
comment = models.CharField(\_(u'Comment'), max\_length=100, blank=True)

### Steel SQUAD

STEEL\_API\_SQUAD\_FIELDS = API\_SQUAD\_FIELDS
STEEL\_API\_SQUAD\_EXTRAS = API\_SQUAD\_EXTRAS
STEEL\_API\_SQUAD\_EXCLUDES = API\_SQUAD\_EXCLUDES

event = models.ForeignKey(SteelMatch, verbose\_name=\_(u'Match'))

### Steel Team & Team Member

Not applicable

# SASS / CAS Matches

SASS / CAS API

## URLs

/sass/match/<key>/ [GET]

/sass/match/<key>/all/ [GET]

/ipsc/match/<key>/stats/ [GET]

/sass/stage/<key>/ [GET]

/sass/competitor/<key>/ [GET]

/sass/competitor/<key>/all/ [GET]

/sass/competitor/<key>/set\_pin/ [POST]

/sass/competitor/<key>/reset\_pin/ [GET]

/sass/competitor/<key>/verify\_all/ [POST]

/sass/scoresheet/<key>/ [GET DELETE]

/sass/scoresheet/<key>/verify/ [POST]

/sass/scoresheet/<key>/unverify/ [GET PUT POST]

/sass/squad/<key>/ [GET]

/sass/match/<key>/competitors/ [GET]

/sass/match/<key>/stages/ [GET]

/sass/match/<key>/squads/ [GET]

/sass/match/<key>/scoresheets/ [POST GET PUT]

/sass/stage/<key>/scoresheets/ [GET]

/sass/competitor/<key>/scoresheets [GET]

## Creating or updating score sheets

### Verify and Unverify

Only the competitor can verify his or her own scorecards. This can be done in two ways; first you can POST to the verify URL for your own scorecards and as you are the competitor for the scorecard it will be verified, secondly anyone can POST to the same URL with a form with field and value ‘password’ for the competitor associated with this scorecard. This last allows for an RO to be logged in and authenticate and have competitor enter their password as verification in client, then POST is done and authenticated for RO and as RO is not competitor the API will look for a password in the form and check that against the competitor and if a match – mark as verified by the competitors him- / her-self. A well-behaving client will not log the competitor password in any manner.

Un-verification is only allowed by under RBAC role.

## Model, fields, extras and excluded

### SASS Match

### SASS Squad (posse)

### SASS Competitor

### SASS Stage

### SASS Scorecard (aka Score Sheet)

### SASS Team

### SASS Member

# NORDIC Matches

Nordic are ‘fältskytte’ in various forms. The field definitions show you the exact choices that are in use and allowed.

## URLs

/nordic/match/<key>/ [GET]

/nordic/match/<key>/all/ [GET]

/nordic/match/<key>/stats/ [GET]

/nordic/stage/<key>/ [GET]

/nordic/competitor/<key>/ [GET]

/nordic/competitor/<key>/all/ [GET]

/nordic/competitor/<key>/set\_pin/ [POST]

/nordic/competitor/<key>/reset\_pin/ [GET]

/nordic/competitor/<key>/verify\_all/ [POST]

/nordic/scoresheet/<key>/ [GET DELETE]

/nordic/scoresheet/<key>/verify/ [POST]

/nordic/scoresheet/<key>/unverify/ [GET PUT POST]

/nordic/squad/<key>/ [GET]

/nordic/match/<key>/competitors/ [GET]

/nordic/match/<key>/stages/ [GET]

/nordic/match/<key>/squads/ [GET]

/nordic/match/<key>/scoresheets/ [POST GET PUT]

/nordic/stage/<key>/scoresheets/ [GET]

/nordic/competitor/<key>/scoresheets [GET]

Note <key> in all above expresses the primary key of the instance accessed.

## Creating or updating score sheets

Each POST or PUT, i.e. create or update of a score sheet must contain all the fields below. PK shall only be expressed in the case you are updating an existing score sheet, if creating a new ensure you provide pk for stage and competitor.

The array can contain one or more score sheets – if you post/put multiple then all will be checked and if and only if all are ok will they be updated or created.

 [

 {

 "comment": "",

 "hits": 3,

 "competitor": 2,

 "precision\_points": 0,

 "created": "2014-02-20T11:41:59Z",

 "is\_locked": false,

 "inner\_hits": 0,

 "updated": "2014-02-27T12:06:23Z",

 "warning": true,

 "dq\_reason": "no",

 "ct\_pk": "85",

 "targets\_hit": 2,

 "pk": 2,

 "model": "match\_nordic.nordicscorecard",

 "is\_verified": false,

 "stage": 1

 }

 {

 "pk": 2,

 "model": "match\_ipsc.ipscscorecard",

 ...

 }

]

POST or PUT for scorecard must contain the following fields and these are the only ones that will be considered;competitor', 'hits', 'inner\_hits', 'targets\_hit', 'precision\_points', 'dq\_reason', 'warning', 'comment'

'stage\_not\_fired'.

If any field is not applicable then set to default value.

### Validation

The following are cross-field validations that will occur on server to validate field interaction. Might be good to check before posting.

* For Field / Field Points:
	+ Ddd
	+

 def clean\_targets\_hit(self):

 if self.cleaned\_data['targets\_hit'] > 6 and (self.instance.stage.event.is\_handgun\_field or self.instance.stage.event.is\_handgun\_field\_points):

 raise forms.ValidationError(\_('Can not hit more then 6 targets with 6 shots'))

 elif self.cleaned\_data['targets\_hit'] > self.instance.stage.number\_of\_targets:

 raise forms.ValidationError(\_('Can not hit more targets then there are in targets in stage (max %s)' % self.instance.stage.number\_of\_targets))

 return self.cleaned\_data['targets\_hit']

 def clean\_hits(self):

 if (self.instance.stage.event.is\_handgun\_field or self.instance.stage.event.is\_handgun\_field\_points) and self.cleaned\_data['hits'] > 6:

 raise forms.ValidationError(\_('Can not register more then 6 hits'))

 return self.cleaned\_data['hits']

 def clean(self):

 if self.instance.stage.event.is\_handgun\_field or self.instance.stage.event.is\_handgun\_field\_points:

 if ((self.cleaned\_data['hits'] > 0 and self.cleaned\_data['targets\_hit'] == 0) or

 (self.cleaned\_data['hits'] == 0 and self.cleaned\_data['targets\_hit'] > 0) or

 (self.cleaned\_data['hits'] > 0 and self.cleaned\_data['targets\_hit'] == 0) or

 (self.cleaned\_data['precision\_points'] > 0 and (self.cleaned\_data['hits'] == 0 and self.cleaned\_data['targets\_hit'] == 0))):

 raise forms.ValidationError(\_('Impossible to score this combination of hits and targets hit'))

 return self.cleaned\_data

## Model, fields, extras and excluded

### NORDIC Match 🡨Match

NORDIC\_MATCH\_LEVELS = (

 ('l1', u'Club'),

 ('l2', u'Group'),

 ('l3', u'Regional'),

 ('l4', u'Nationals'),

 ('l5', u'Championships'),

 ('l6', u'International'),

 ('lx', u'Not Sanctioned'),

)

NORDIC\_MEDAL = (

 ('-', u'-'),

 ('B', u'Bronze'),

 ('S', u'Silver'),

 ('G', u'Gold'),

)

NORDIC\_HANDGUN\_WC = (

 ('HA1', u'A1'),

 ('HA2', u'A2'),

 ('HA3', u'A3'),

 ('HB1', u'B1'),

 ('HB2', u'B2'),

 ('HB3', u'B3'),

 ('HC1', u'C1'),

 ('HC2', u'C2'),

 ('HC3', u'C3'),

 ('HD1', u'D1'),

 ('HD2', u'D2'),

 ('HD3', u'D3'),

 ('HR1', u'R1'),

 ('HR2', u'R2'),

 ('HR3', u'R3'),

 ('HJ', u'Junior'),

 ('HVY', u'Veteran Young'),

 ('HVO', u'Veteran Old'),

)

NORDIC\_HANDGUN\_WC\_CLASS\_GROUP = {

 'HA1': ('A', '1'),

 'HA2': ('A', '2'),

 'HA3': ('A', '3'),

 'HB1': ('B', '1'),

 'HB2': ('B', '2'),

 'HB3': ('B', '3'),

 'HC1': ('C', '1'),

 'HC2': ('C', '2'),

 'HC3': ('C', '3'),

 'HD1': ('D', '1'),

 'HD2': ('D', '2'),

 'HD3': ('D', '3'),

 'HR1': ('R', '1'),

 'HR2': ('R', '2'),

 'HR3': ('R' ,'3'),

 'HJ': ('C', 'Junior'),

 'HVY': ('C', 'Veteran Young'),

 'HVO': ('C', 'Veteran Old'),

}

NORDIC\_MAGNUM\_WC = (

 ('MM1', u'M1'),

 ('MM2', u'M2'),

 ('MM3', u'M3'),

 ('MM4', u'M4'),

 ('MM5', u'M5'),

 ('MM6', u'M6'),

 ('MM7', u'M7'),

 ('MM8', u'M8'),

)

NORDIC\_PRECISION\_WC = (

 ('PP1', u'P1'),

 ('PP2', u'P1'),

 ('PP3', u'P1'),

 ('PP1', u'P1'),

 ('PP2', u'P2'),

 ('PP3', u'P3'),

)

NORDIC\_RIFLE\_WC = (

 ('RG1', u'G1'),

 ('RG2', u'G2'),

 ('RG3', u'G3'),

 ('RG1', u'G4'),

 ('RG2', u'G5'),

 ('RG3', u'G6'),

)

NORDIC\_WC = NORDIC\_HANDGUN\_WC + NORDIC\_MAGNUM\_WC + NORDIC\_RIFLE\_WC + NORDIC\_PRECISION\_WC

NORDIC\_SQUAD\_WC = (('xxx', u'Any'),) + NORDIC\_WC

NORDIC\_DQ\_REASON = (

 ('no', u'Not disqualified'),

 ('ug', u'Unsafe gun handling'),

 ('uc', u'Unsportsmanlike conduct'),

 ('sc', u'Violations of the Shooter Code of Conduct'),

 ('am', u'Ammunition fails to meet requirements'),

 ('wp', u'Weapon fails to meet requirements'),

 ('ad', u'Accidental discharge'),

 ('gr', u'General regulations'),

 ('mw', u'Multiple warnings'),

 ('pc', u'Prohibited substances'),

)

NORDIC\_MATCH\_RULE = ( ('hf', u'Handgun Field'),

 ('pf', u'Handgun Field Points'),

 ('mf', u'Magnum Field'),

 ('pr', u'Precision Handgun'),

 ('rf', u'Rifle Field '),)

NORDIC\_API\_MATCH\_FIELDS = match.models.API\_MATCH\_FIELDS + ( 'level',

 sub\_rule',

 weapon\_classes',

 merge\_1\_3',

 precision\_series',

 precision\_shots\_per\_series',)

NORDIC\_API\_MATCH\_EXTRAS = match.models.API\_MATCH\_EXTRAS + ( 'minimum\_rounds',

 number\_of\_stages',)

NORDIC\_API\_MATCH\_EXCLUDES = match.models.API\_MATCH\_EXCLUDES

# ---------------------------------------------------------------------------

class NordicMatch(match.models.Match):

 level = models.CharField(\_(u'Level'), max\_length=2, choices=NORDIC\_MATCH\_LEVELS, default='l1')

 sub\_rule = models.CharField(\_(u'Sub rule'), max\_length=3, choices=NORDIC\_MATCH\_RULE, default='hf')

 weapon\_classes = MultiSelectField(\_(u'Recognized Weapon & Classes'), max\_length=200, choices=NORDIC\_WC, default='A')

 merge\_1\_3 = models.BooleanField(\_(u'Combined results and standard medals'), default=False) # shall standard medals be generated and combined be shown

 precision\_series = models.PositiveIntegerField(\_('Precision Series'), default=4)

 precision\_shots\_per\_series = models.PositiveIntegerField(\_('Shots per serie'), default=5)

 \_scoring\_completed = models.PositiveIntegerField(\_('Scoring completed'), default=0, editable=False)

 merchandizes = generic.GenericRelation(merchandize.Merchandize2)

### NORDIC Squad 🡨 squad

NORDIC\_API\_SQUAD\_FIELDS = match.models.API\_SQUAD\_FIELDS + ( 'weapon\_classes',)
NORDIC\_API\_SQUAD\_EXTRAS = match.models.API\_SQUAD\_EXTRAS + ( 'starts',
 stops', )

NORDIC\_API\_SQUAD\_EXCLUDES = match.models.API\_SQUAD\_EXCLUDES

class NordicSquad(match.models.Squad):

 event = models.ForeignKey(NordicMatch, verbose\_name=\_(u'Match'))
 weapon\_classes = MultiSelectField(\_(u'Allowed Weapon & Classes'), max\_length=200, choices=NORDIC\_SQUAD\_WC, default='-', blank=True)
 \_starts = models.DateTimeField(\_(u'Start'), blank=True, null=True, editable=True)
 \_stops = models.DateTimeField(\_(u'Finish'), blank=True, null=True, editable=True)

### NORDIC Competitor 🡨 competitor

NORDIC\_EMPTY\_STRING = '0,0,0,0,0,0,0' # X 10 9 8 7 6 5 4 3 2 1 0 M

NORDIC\_API\_COMPETITOR\_FIELDS = match.models.API\_COMPETITOR\_FIELDS + ( 'squad',

 weapon\_class',

 nordic\_num',

 s1',

 s2',

 s3',

 s4',

 s5',

 s6',

 s7',

 s8',

 s9',

 s10',

 s11',

 s12',

 warning',)

NORDIC\_API\_COMPETITOR\_EXTRAS = match.models.API\_COMPETITOR\_EXTRAS

NORDIC\_API\_COMPETITOR\_EXCLUDES = match.models.API\_COMPETITOR\_EXCLUDES

class NordicCompetitor(match.models.Competitor):

 event = models.ForeignKey(NordicMatch, verbose\_name=\_(u'Match'))

 squad = models.ForeignKey(NordicSquad, verbose\_name=\_(u'Squad'), blank=True, null=True)

 weapon\_class = models.CharField(\_(u'Weapon & Class'), max\_length=4, choices=NORDIC\_WC, default='l1')

 nordic\_num = models.CharField(\_(u'Nordic num'), max\_length=16, blank=True) # NORDIC.org license number

 \_place = models.PositiveIntegerField(verbose\_name=\_(u'Place'), default=0)

 \_combined\_place = models.PositiveIntegerField(verbose\_name=\_(u'Combined Place'), default=0)

 \_tot\_hits = models.PositiveIntegerField(\_(u'Total hits'), default=0)

 \_tot\_inner\_hits = models.PositiveIntegerField(\_(u'Total inner hits'), default=0)

 \_tot\_targets\_hit = models.PositiveIntegerField(\_(u'Total targets hit'), default=0)

 \_tot\_precision\_points = models.PositiveIntegerField(\_(u'Total precision points'), default=0)

 \_std\_medal = models.CharField(\_(u'Standard medal'), max\_length=2, choices=NORDIC\_MEDAL, default='-') # RENAME, USE DIRECTLY

 # For field precision only, each serie is X 10 9 8 7 6 5 4 3 2 1 0 M (10+ values) per target registered

 s1 = models.CommaSeparatedIntegerField(\_(u'String 1'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s2 = models.CommaSeparatedIntegerField(\_(u'String 2'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s3 = models.CommaSeparatedIntegerField(\_(u'String 3'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s4 = models.CommaSeparatedIntegerField(\_(u'String 4'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s5 = models.CommaSeparatedIntegerField(\_(u'String 5'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s6 = models.CommaSeparatedIntegerField(\_(u'String 6'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s7 = models.CommaSeparatedIntegerField(\_(u'String 7'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s8 = models.CommaSeparatedIntegerField(\_(u'String 8'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s9 = models.CommaSeparatedIntegerField(\_(u'String 9'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s10 = models.CommaSeparatedIntegerField(\_(u'String 10'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s11 = models.CommaSeparatedIntegerField(\_(u'String 11'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 s12 = models.CommaSeparatedIntegerField(\_(u'String 12'), max\_length=70, default=NORDIC\_EMPTY\_STRING)

 warning = models.BooleanField(\_(u'Warning'), default=False)

 # fixme -- need to add DQ settings here as we might not always use scorecards

### NORDIC Stage 🡨 stage

NORDIC\_API\_STAGE\_FIELDS = match.models.API\_STAGE\_FIELDS + ( 'event',

 targets',

 inner\_targets',

 falling\_targets',

 precision\_targets',

 rounds',

 scoring',

 time',

 time2',

 time3',

 time4',

 # 'start\_pos', already in stage

 start\_pos2',

 start\_pos3',

 start\_pos4',

 groups',)

NORDIC\_API\_STAGE\_EXTRAS = match.models.API\_STAGE\_EXTRAS

NORDIC\_API\_STAGE\_EXCLUDES = match.models.API\_STAGE\_EXCLUDES

NORDIC\_STAGE\_SCORING = (('n', u'Normal'),

 ('s', u'Shoot-off'),)

class NordicStage(match.models.Stage):

 event = models.ForeignKey(NordicMatch, verbose\_name=\_(u'Match'))

 targets = models.PositiveIntegerField(\_('Targets'), default=0)

 inner\_targets = models.PositiveIntegerField(\_('Targets w. inner'), default=0)

 falling\_targets = models.PositiveIntegerField(\_('Falling Targets'), default=0)

 precision\_targets = models.PositiveIntegerField(\_('Precision targets'), default=0)

 rounds = models.PositiveIntegerField(\_('Rounds'), default=6)

 scoring = models.CharField(\_(u'Scoring'), max\_length=4, choices=NORDIC\_STAGE\_SCORING, default='n')

 time = models.DecimalField(verbose\_name=\_(u'A Time (sec)'), max\_digits=5, decimal\_places=0, default=0)

 time2 = models.DecimalField(verbose\_name=\_(u'B Time (sec)'), max\_digits=5, decimal\_places=0, default=0)

 time3 = models.DecimalField(verbose\_name=\_(u'C Time (sec)'), max\_digits=5, decimal\_places=0, default=0)

 time4 = models.DecimalField(verbose\_name=\_(u'R Time (sec)'), max\_digits=5, decimal\_places=0, default=0)

 start\_pos2 = models.CharField(\_(u'B Start position'), default='', max\_length=100, blank=True)

 start\_pos3 = models.CharField(\_(u'C Start position'), default='', max\_length=100, blank=True)

 start\_pos4 = models.CharField(\_(u'R Start position'), default='', max\_length=100, blank=True)

 groups = models.PositiveIntegerField(\_('Target groups'), default=1)

 \_scoring\_completed = models.PositiveIntegerField(\_('Scoring completed'), default=0, editable=False)

### NORDIC Scorecard (aka Score Sheet) 🡨 scorecard

NORDIC\_API\_SCORECARD\_FIELDS = match.models.API\_SCORECARD\_FIELDS + ( 'stage',

 competitor',

 hits',

 inner\_hits',

 targets\_hit',

 precision\_points',

 dq\_reason',

 warning',

 comment',)

NORDIC\_API\_SCORECARD\_EXTRAS = match.models.API\_SCORECARD\_EXTRAS

NORDIC\_API\_SCORECARD\_EXCLUDES = match.models.API\_SCORECARD\_EXCLUDES + ('comment',)

class NordicScoreCard(match.models.ScoreCard):

 stage = models.ForeignKey(NordicStage, verbose\_name=\_(u'Stage'))

 competitor = models.ForeignKey(NordicCompetitor, verbose\_name=\_(u'Competitor'))

 hits = models.PositiveIntegerField(\_(u'#Hits'), default=0)

 inner\_hits = models.PositiveIntegerField(\_(u'#Inner hits'), default=0)

 targets\_hit = models.PositiveIntegerField(\_(u'#Targets hit'), default=0)

 precision\_points = models.PositiveIntegerField(\_(u'#Points'), default=0)

 dq\_reason = models.CharField(\_(u'DQ reason'), max\_length=2, choices=NORDIC\_DQ\_REASON, default='no')

 warning = models.BooleanField(\_(u'Warning'), default=False)

 comment = models.CharField(\_(u'Comment'), max\_length=100, blank=True)

### NORDIC Team 🡨 team

Not supported yet.

### NORDIC Team Member 🡨 team member

Not supported yet.

Generic Fields, Extras and Excluded Data

Below are details for the generic Event or Match, Competitor, Stage, Team, Team member and Squad. This information is the minimal information in common for all these.

### Match or Serie

MATCH\_REGISTRATION = ( ('op', \_(u'open, all approved directly')),

 ('ap', \_(u'open, require approval manually')),

 ('ow', \_(u'open, all competitors placed on waiting list')),
 ('cl', \_(u'closed')))

# Draft --- match is not shown to outside, only to creator

# Active --- match is ongoing and shown outside account if public

# Completed --- match is completed and no results can be registered

# Cancelled -- match was cancelled

MATCH\_STATUS = (('dr', \_(u'draft')),

 ('on', \_(u'active')),

 ('ol', \_(u'active, no self-edit')),
 ('pr', \_(u'preliminary completed')),
 ('cp', \_(u'completed')),
 ('cs', \_(u'cancelled')))

PREMATCH\_STATUS = ( ('no', \_(u'not available')),

 ('oi', \_(u'available and included in results in main-match')),

 ('oe', \_(u'available but excluded in results from main-match')))

MATCH\_RULE = ( ('pp', u'PPC'),

 ('ip', u'IPSC'),

 ('id', u'IDPA'),

 ('us', u'USPSA'),

 ('nd', u'Nordic'),

 ('sc', u'Steel'))

MATCH\_RESULTS = ( ('all', u'results and scores are shown to anyone'),

 ('stg', u'scores shown to anyone but results only shown to organizers'),

 ('org', u'result and scores only shown to organizers'))

# API fields & extra fields shown for any match/event

API\_MATCH\_FIELDS = ( 'name',

 description',

 max\_competitors',

 prematch',

 max\_prematch\_competitors',

 allow\_teams',

 status',

 registration',

 results',

 rule',

 region',

 currency',

 url',

 url\_display',

 multiple\_reg\_allowed',

 lat',

 lng' )

API\_MATCH\_EXTRAS = ('role\_names', '

 starts',

 'ends',

 'registration\_closes',

 'registration\_starts',

 'created',

 'updated',

 'is\_locked',

 ‘short\_url’,

 number\_of\_mainmatch\_competitors\_approved',

 number\_of\_mainmatch\_competitors\_pending',

 number\_of\_mainmatch\_competitors\_waiting',

 number\_of\_prematch\_competitors\_approved',

 number\_of\_prematch\_competitors\_pending',

 number\_of\_prematch\_competitors\_waiting',)

API\_MATCH\_EXCLUDES = ('created', 'updated', 'is\_locked')

**Fields:**

name = models.CharField(\_(u'Name'), max\_length=40)

description = models.CharField(\_(u'Description'), max\_length=300, blank=True)

max\_competitors = models.PositiveIntegerField(\_('Max competitors'), default=100) # max competitors in match or main match

prematch = models.CharField(\_(u'Pre-match'), max\_length=2, choices=PREMATCH\_STATUS, default='no') # do we have a prematch, and is scoring included

max\_prematch\_competitors = models.PositiveIntegerField(\_('Max pre-match competitors'), default=0)

allow\_teams = models.BooleanField(\_(u'Allow teams'), default=False)

registration = models.CharField(\_(u'Registration is'), max\_length=2, choices=MATCH\_REGISTRATION, default='op')

results = models.CharField(\_(u'Results are'), max\_length=3, choices=MATCH\_RESULTS, default='all')

rule = models.CharField(\_(u'Type'), max\_length=2, choices=MATCH\_RULE, default='ip', editable=False)

region = models.CharField(\_(u'Region'), max\_length=3, choices=REGIONS\_3166A3, default='---')

currency = models.CharField(\_(u'Currency'), max\_length=3, choices=CURRENCIES\_ISO\_4217, default='EUR')

url = models.URLField(\_(u'Web link (http://...)'), verify\_exists=True, blank=True)

url\_display = models.CharField(\_(u'Web link text'), blank=True, max\_length=20)

multiple\_reg\_allowed = models.BooleanField(\_(u'Multiple registrations allowed'), default=False) # can a shooter register more then once

public = models.BooleanField(\_(u'Public'), default=True) # it this match publicly visisble

state = models.CharField(\_(u'State'), blank=True, max\_length=3, choices=US\_STATE\_CHOICES)

lat = models.DecimalField(\_(u'Latitude'), max\_digits=10, decimal\_places=5, blank=True, null=True)

lng = models.DecimalField(\_(u'Longitude'), max\_digits=10, decimal\_places=5, blank=True, null=True)

**Extras**

 @localdatetime('\_starts')

 def starts(self):

 # returns start time given current user

 return get\_current\_tz()

 @localdatetime('\_ends')

 def ends(self):

 # returns end time given current user

 return get\_current\_tz()

 @localdatetime('\_registration\_closes')

 def registration\_closes(self):

 # returns reg closes given current user

 return get\_current\_tz()

 @property

 def short\_url(self):

 '''returns short url for event, currently using bit.ly service'''

### Competitor

COMPETITOR\_STATUS = ( ('w', \_(u'Waiting list')),

 ('p', \_(u'Pending')),

 ('a', \_(u'Approved')),

 ('d', \_(u'Declined')),

 ('x', \_(u'Deleted')))

COMPETITOR\_CODE = ( ('-', \_(u'-')),

 ('VIP', \_(u'VIP')),

 ('Sp1', \_(u'Sponsor 1 (Sp1)')),

 ('Sp2', \_(u'Sponsor 2 (Sp2)')),

 ('Sp3', \_(u'Sponsor 3 (Sp3)')),

 ('St1', \_(u'Staff 1 (St1)')),

 ('St2', \_(u'Staff 2 (St2)')),

 ('St3', \_(u'Staff 3 (St3)')),

 ('Cp1', \_(u'Competitor 1 (Cp1)')),

 ('Cp2', \_(u'Competitor 2 (Cp2)')),

 ('Cp3', \_(u'Competitor 2 (Cp3)')),

 )

# API fields & extra fields shown for any match/event

API\_COMPETITOR\_FIELDS = ( 'shooter',

 first\_name',

 last\_name',

 sex',

 email',

 region',

 club',

 number',

 status',

 code',

 comment',

 prematch',

 license')

API\_COMPETITOR\_EXTRAS = ('get\_phone\_display',

 'is\_paid',

 'created',

 'updated',

 'is\_locked',)

API\_COMPETITOR\_EXCLUDES = ('get\_phone\_display',

 'is\_paid',

 'email',

 'comment',

 'license',

 'created',

 'updated',

 'is\_locked',

 ‘code’)

### Fields

shooter = models.ForeignKey(User, verbose\_name=\_(u'Shooter'))

 '''

 the first, last, sex, email are not editable within a competitor, they are directly copied

 in from the shooter (see save of CompetitorForm)

 the rest of the fields are preset when registering but also editable for a competitor

 this logic could possibly be revised, still need to ensure name remains the same for a competitor

 even if shooter changes name/sex

 '''

first\_name = models.CharField(\_(u'First name'), max\_length=30, blank=True)

last\_name = models.CharField(\_(u'Last name'), max\_length=30, blank=True)

sex = models.CharField(\_(u'Sex'), max\_length=1, choices=SEX\_CHOICES, default='m')

email = models.EmailField(\_('Email'), max\_length=300, blank=True)

region = models.CharField(\_(u'Region'), max\_length=3, choices=REGIONS\_3166A3, default='---')

club = models.CharField(\_(u'Club'), max\_length=50, blank=True)

\_phone = models.CharField(\_(u'Phone'), max\_length=30, blank=True)

number = models.PositiveIntegerField(\_('Number'))

status = models.CharField(\_(u'Registration'), max\_length=1, choices=COMPETITOR\_STATUS, default='a')

code = models.CharField(\_(u'Org code'), max\_length=5, choices=COMPETITOR\_CODE, default='-') # internal, for organizers only

comment = models.CharField(\_(u'Org comment'), max\_length=300, blank=True) # internal, for organizers only

prematch = models.BooleanField(\_(u'Prematch'), default=False)

license = models.CharField(\_(u'License No'), max\_length=30, blank=True)

**Extras**

 @property

 def get\_phone\_display(self):

 # FIXME -- add smart stuff to make this telephonenumber callable

 if self.\_phone != '':

 data = self.\_phone.split('-', 1)

 local = data[1][1:] if data[1][0]=='0' else data[1] # remove leading 0

 return '%s%s' % (data[0], local)

 else:

 return '-'

 @property

 def is\_locked(self):

 return self.\_locked

 @localdatetime('\_lock\_changed')

 def lock\_changed(self):

 return get\_current\_tz()

 @localdatetime('\_created')

 def created(self):

 return get\_current\_tz()

 @localdatetime('\_updated')

 def updated(self):

 return get\_current\_tz()

### Squad

SQUAD\_REGISTRATION = (('aa', \_(u'Anyone')), ('os', \_(u'Restricted')))

# API fields & extra fields shown for any match/event

API\_SQUAD\_FIELDS = ( 'event', 'number',

 max\_competitors',

 comment',

 registration',

 prematch')

API\_SQUAD\_EXTRAS = ('is\_full', 'created', 'updated', 'is\_locked')

API\_SQUAD\_EXCLUDES = ('comment', 'created', 'updated', 'is\_locked')

**Fields**

number = models.PositiveIntegerField(\_('Number'))

max\_competitors = models.PositiveIntegerField(\_('Max competitors'), default=10)

comment = models.CharField(\_(u'Comment'), max\_length=100, blank=True)

registration = models.CharField(\_(u'Registration'), max\_length=2, choices=SQUAD\_REGISTRATION, default='aa')

prematch = models.BooleanField(\_(u'Pre-match only'), default=False)

**Extras**

 @localdatetime('\_created')

 def created(self):

 return get\_current\_tz()

 @localdatetime('\_updated')

 def updated(self):

 return get\_current\_tz()

 @property

 def is\_locked(self):

 return self.\_locked

### Team

TEAM\_STATUS = ( ('p', \_(u'Pending')),

 ('a', \_(u'Approved')),

 ('d', \_(u'Declined'))

 )

TEAM\_REGION = (('---', 'not specified, allow any region'),) + REGIONS\_3166A3

# API fields & extra fields shown for any match/event

API\_TEAM\_FIELDS = ('number',

 name',

 region',

 status',

 paid',

 comment')

API\_TEAM\_EXTRAS = ('created', 'updated', 'is\_locked')

API\_TEAM\_EXCLUDES = ('comment', 'paid', 'created', 'updated', 'is\_locked')

class Team(models.Model):

**Fields**

number = models.PositiveIntegerField(\_('Number'))

name = models.CharField(\_(u'Name'), max\_length=45, blank=True)

region = models.CharField(\_(u'Region'), max\_length=3, choices=TEAM\_REGION, default='---')

status = models.CharField(\_(u'Status'), max\_length=1, choices=TEAM\_STATUS, default='p')

paid = models.BooleanField(\_(u'Paid'), default=False)

comment = models.CharField(\_(u'Org comment'), max\_length=300, blank=True) # internal, for organizers only

**Extras**

 @localdatetime('\_created')

 def created(self):

 return get\_current\_tz()

 @localdatetime('\_updated')

 def updated(self):

 return get\_current\_tz()

 @property

 def is\_locked(self):

 return self.\_locked

### Team member

MEMBERSHIP\_STATUS = (('ac', \_(u'Accepted')), ('pe', \_(u'Pending')), ('rj', \_(u'Rejected')))

# API fields & extra fields shown for any match/event

API\_TEAMMEMBER\_FIELDS = ( 'status',

 comment',

 team',

 competitor')

API\_TEAMMEMBER\_EXTRAS = ('created', 'updated', 'is\_locked')

API\_TEAMMEMBER\_EXCLUDES = ('comment', 'status', 'created', 'updated', 'is\_locked')

class TeamMember(models.Model):

# competitor = THE EVENT COMPETITOR

# team = the team

status = models.CharField(\_(u'Status'), max\_length=2, choices=MEMBERSHIP\_STATUS, default='pe')

comment = models.CharField(\_(u'Org comment'), max\_length=300, blank=True) # internal, for organizers only

 @localdatetime('\_created')

 def created(self):

 return get\_current\_tz()

 @localdatetime('\_updated')

 def updated(self):

 return get\_current\_tz()

 @property

 def is\_locked(self):

 return self.\_locked

### Stage

# API fields & extra fields shown for any match/event

API\_STAGE\_FIELDS = ( 'number',

 name',

 included')
API\_STAGE\_EXTRAS = ('created', 'updated', 'is\_locked')

API\_STAGE\_EXCLUDES = ('created', 'updated', 'is\_locked')

number = models.PositiveIntegerField(\_('Number'))

name = models.CharField(\_(u'Name'), max\_length=30, blank=True)

# Groups and Match Officials

Each event within SSI has a ‘group’ (permanent as in can be used for many events or standalones as in used for only one event). The group is used to manage authorization and manage the organizers of an event or group of events (who is ‘admin’, ‘staff’ or ‘member’).

The group also allows you to assign match official roles within the event(s) e.g. match director, range officer etc.

For full details on authorization and match officials see the ‘SSI Reference Manual’.

## Statistics URL

To access this information and e.g. generate reporting to various organizations each sport has a URL that allows fetching these statistics:

/idpa/match/<key>/stats/ [GET]

/ipsc/match/<key>/stats/ [GET]

/ppc/match/<key>/stats/ [GET]

/sass/match/<key>/stats/ [GET]

/steel/match/<key>/stats/ [GET]

## JSON Format

A request to any of the statistic URLs will return a JSON body (list) consisting of; event details, group details and match officials information.

Each match official consists of the official role abbreviation and a list of all match officials where each with their name, pk and other details. See example below:

 [

 {

 "handgun\_divs": "hg1,hg2,hg3,hg5,hg12",

 "number\_of\_prematch\_competitors\_approved": 0,

 "short\_url": "http://bit.ly/zElgSO",

 ...

 "allow\_teams": false

 }

,

 {

 "name": "lkjaelkj gröp",

 "permanent": true,

 "ct\_pk": "18",

 "active": true,

 "pk": 1,

 "model": "shootgroups.shootgroup",

 "public": true,

 "description": "df fs dfsdf "

 }

,

 {

 "RO": [

 {

 "first\_name": "Jöns",

 "last\_name": "Lundström",

 "tel": "-",

 "waid": "",

 "idpa\_num": "90980989",

 "pk": 1,

 "iroa\_num": "12345",

 "ics\_alias": "",

 "uspsa\_num": "",

 "email": "jens@lundstroem.com"

 },

 {

 "first\_name": "Pete",

 "last\_name": "Blöw",

 "tel": "-",

 "iroa\_num": "12343",

 "waid": "",

 "idpa\_num": "",

 "pk": 4,

 "ics\_alias": "",

 "uspsa\_num": "",

 "email": "pete@lundstroem.com"

 }

 ]

 },

 {

 "CO": []

 },

 {

 "CRO": [

 {

 "first\_name": "Johan",

 "last\_name": "Lisebörn",

 "tel": "-",

 "waid": "",

 "idpa\_num": "",

 "pk": 7,

 "iroa\_num": "12347",

 "ics\_alias": "",

 "uspsa\_num": "",

 "email": "johaaan@lundstroem.com"

 }

 ]

 },

 {

 "SF": []

 },

 {

 "CSF": []

 },

 {

 "SO": []

 },

 {

 "CSO": []

 },

 {

 "QM": []

 },

 {

 "RM": []

 },

 {

 "MD": []

 },

 {

 "SD": []

 }

]

For each match official the following information is provided and can be used as needed;

First name, last name, pk (unique id within SSI for this user/shooter), tel (a fully qualified e164 number with country code), waid (WA1500 org id), idpa licens number, ics alias (IPCS), USPSA number, IROA number and email.

For each event the same information as when getting a individual event over API is provided.

For group some basic information is provided.

### Group

API\_GROUP\_FIELDS = ('name',

 description',

 public',

 active',

 permanent',)

API\_GROUP\_EXTRAS = ()

API\_GROUP\_EXCLUDES = ('created', 'updated', 'is\_locked',)

class ShootGroup(models.Model):

 active = models.BooleanField(\_(u'Active'), default=True)

 name = models.CharField(\_(u'Group name'), max\_length=30, db\_index=True)

 description = models.CharField(\_(u'Description'), max\_length=300, blank=True)

 permanent = models.BooleanField(\_(u'Permanent'), default=True) # permanent means it is not only for one match, i.e. is it a standalone (False) or reusable (True)

 public = models.BooleanField(\_(u'Public'), default=True, db\_index=True) # public means that this group is visible when other shooter searches for groups

 created\_by = models.ForeignKey(User, verbose\_name=\_(u'Created by'), related\_name='created\_shootgroup\_set', blank=True, null=True, editable=False)

 \_created = models.DateTimeField(\_(u'Created'), auto\_now\_add=True, editable=False)

 updated\_by = models.ForeignKey(User, verbose\_name=\_(u'Updated by'), related\_name='updated\_shootgroup\_set', blank=True, null=True, editable=False)

 \_updated = models.DateTimeField(\_(u'Updated'), auto\_now=True, editable=False)

### Match Official

The following is the match official roles available and their abbreviations as used. Note that SSI does note enforce any checks or requirements and a shooter can e.g. be both RO and MD. Any enforcing or validation of this is left up to match organizers themselves. (exception – a shooter can not be both RO and CRO, SF and CSF, or ST and CST).

MATCH\_OFFICIAL = ( ('RO', 'Range Officer'), # IPSC, USPSA, STEEL
 ('CO', 'Chrono Officer'), # USPSA
 ('CRO', 'Chief Range Officer'), # IPSC, USPSA
 ('SF', 'Safety Officer'), # IDPA
 ('CSF', 'Chief Safety Officer'), # IDPA own
 ('SO', 'Stats Officer'), # IPSC, USPSA
 ('CSO', 'Chief Stats Officer'), # IPSC own
 ('QM', 'Quarter Master'), # IPSC, USPSA
 ('RM', 'Range Master'), # IPSC, USPSA
 ('MD', 'Match Director'), # IPSC, USPSA, …
 ('SD', 'Series Director'))