
EVE SRP Documentation

Release

Will Ross

October 04, 2017

1	User Guide	3
1.1	Quick Start	3
1.2	External API	14
2	Developers Guide	21
2.1	Authentication	21
2.2	Killmail Handling	30
2.3	Views	35
2.4	Models	38
2.5	Javascript	42
3	Indices and tables	45
	Python Module Index	47

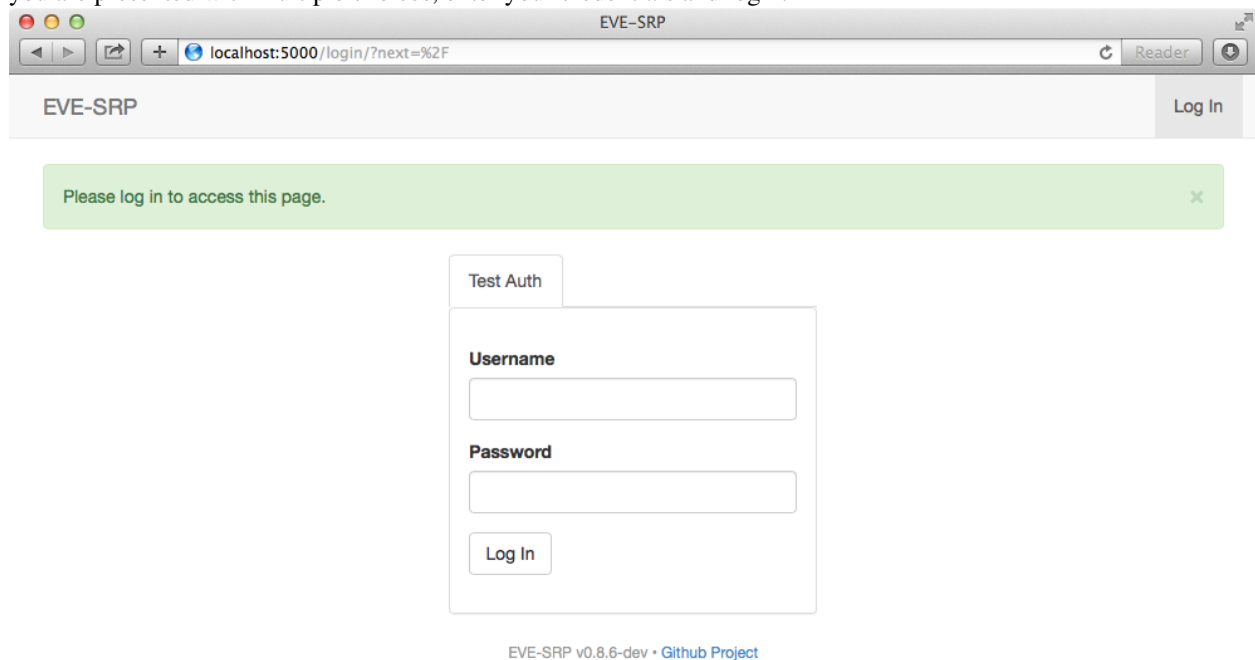
EVE-SRP is designed to facilitate a ship replacement (SRP) or reimbursement program in the game EVE Online. It features a pluggable authentication setup so it can integrate with existing authentication systems, and comes with built in support for TEST Alliance's Auth and Brave's Core systems. It also features a configurable killmail source system, with built in support for zKillboard based killboards and the recent ESI killmail endpoint. Again, this is an extensible system so if you have a custom killboard, as long as there's some sort of programmatic access, you can probably write a custom adapter.

For the users, EVE-SRP offers quick submission and an easy way to check your SRP pending requests. On the administrative side, EVE-SRP uses the concept of divisions, with different users and groups of users being able to submit requests, review them (set payouts and approve or reject requests), and finally pay out approved requests. This separation allows spreading of the labor intensive and low risk task of evaluating requests from the high privilege of paying out requests from a central wallet. This also means different groups can have different reviewing+paying teams. For example, you may wish for capital losses to be reviewed by a special team that is aware of your capital group's fitting requirements, and in lieu of payouts you may have someone hand out replacement hulls.

Quick Start

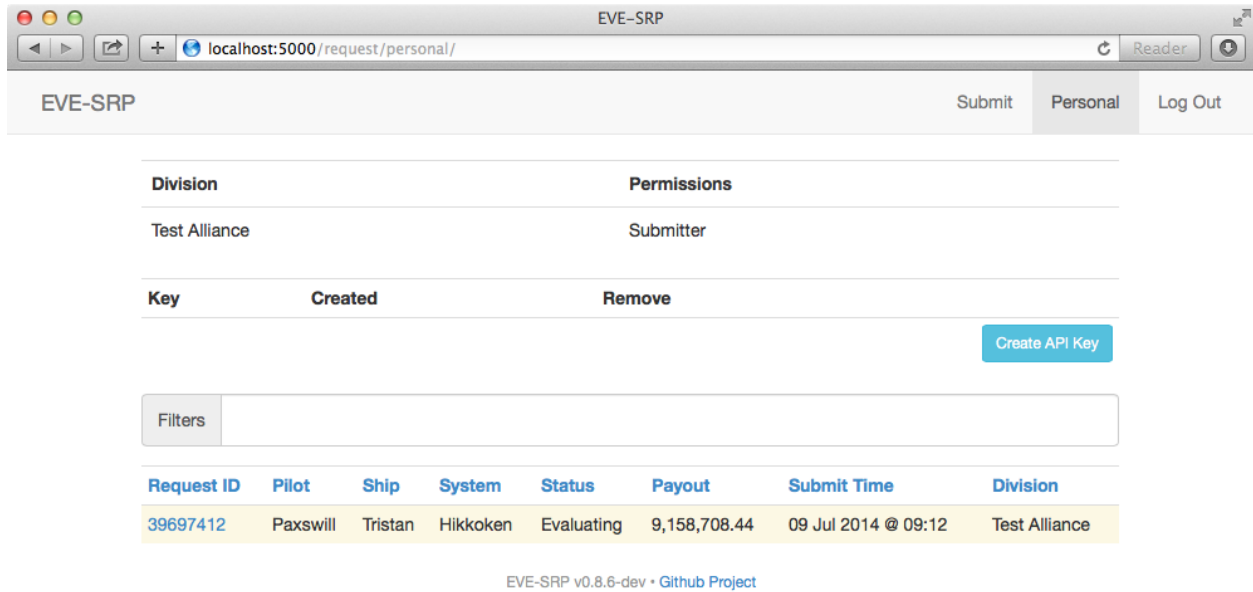
Logging in and Submitting Requests

When you first access a website running EVE-SRP, you will be asked to login. Select the appropriate login option if you are presented with multiple choices, enter your credentials and login.



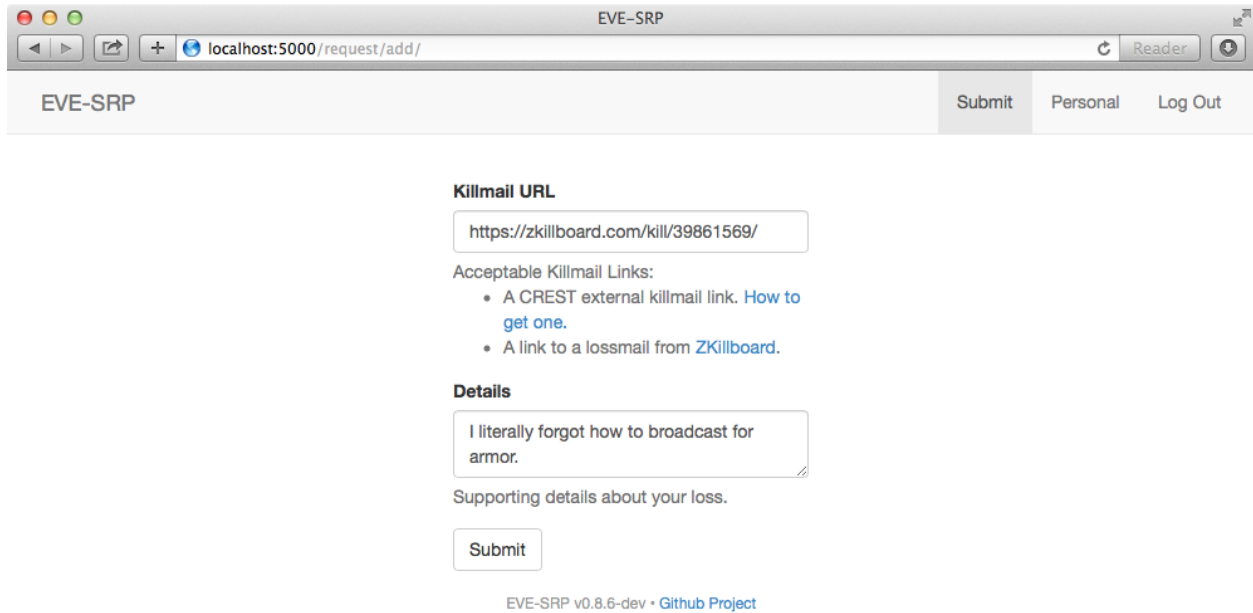
The screenshot shows a web browser window with the title "EVE-SRP". The address bar displays "localhost:5000/login/?next=%2F". The page header includes "EVE-SRP" on the left and a "Log In" button on the right. A green notification bar at the top states "Please log in to access this page." with a close button. The main content area features a "Test Auth" tab and a login form. The form contains two input fields: "Username" and "Password", followed by a "Log In" button. At the bottom of the page, the text "EVE-SRP v0.8.6-dev • [Github Project](#)" is visible.

Once you have logged in, you will be able to see what reimbursement divisions you have been granted permissions in as well as all of the requests you have submitted.



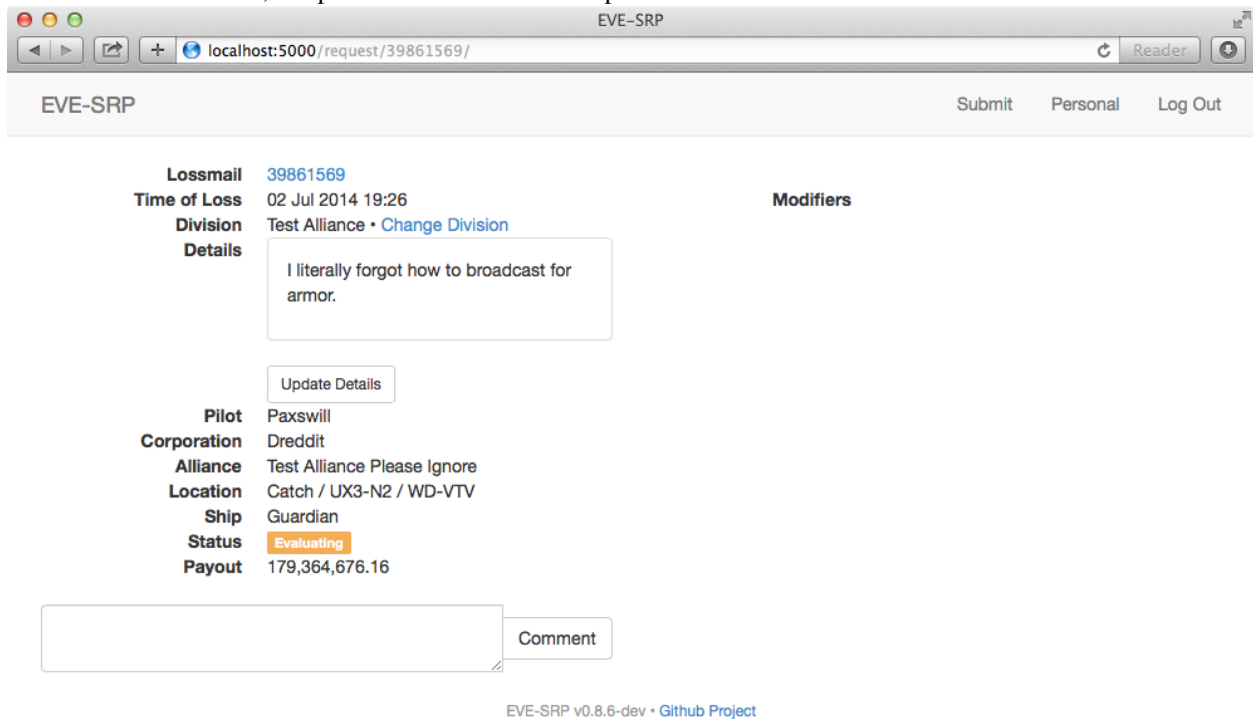
To submit a request, click the “Submit” button at the top of the screen. The button will only be present if you have been granted the submit privilege within a division.

In the form, enter a killmail URL and any details your organization normally requires. What kind of killmail URLs that are acceptable is up to your organization, but common choices are zKillboard based killboards or CREST killmail URLs from an in-game killmail. Click the “Submit” button once you are done entering the information.



The screenshot shows a web browser window titled "EVE-SRP" with the address bar displaying "localhost:5000/request/add/". The page has a header with "EVE-SRP" on the left and "Submit", "Personal", and "Log Out" on the right. The main content area is titled "Killmail URL" and contains a text input field with the value "https://zkillboard.com/kill/39861569/". Below this is a section titled "Acceptable Killmail Links:" with two bullet points: "A CREST external killmail link. [How to get one.](#)" and "A link to a lossmail from [ZKillboard.](#)". Another section titled "Details" contains a text input field with the value "I literally forgot how to broadcast for armor." and a "Submit" button. At the bottom, it says "EVE-SRP v0.8.6-dev • [Github Project](#)".

You will be redirected to the request detail page once you have submitted your request. Via this page you can add comments for reviewers, or update the details to correct problems.



The screenshot shows the "request/39861569/" page in the browser. The header is similar to the previous page. The main content area displays details for a lossmail. On the left, there is a list of fields: "Lossmail" (39861569), "Time of Loss" (02 Jul 2014 19:26), "Division" (Test Alliance • [Change Division](#)), "Details" (I literally forgot how to broadcast for armor.), "Pilot" (Paxswill), "Corporation" (Dreddit), "Alliance" (Test Alliance Please Ignore), "Location" (Catch / UX3-N2 / WD-VTV), "Ship" (Guardian), "Status" (Evaluating), and "Payout" (179,364,676.16). To the right of the "Details" field is an "Update Details" button. Below the "Status" field is a "Comment" button. At the bottom, it says "EVE-SRP v0.8.6-dev • [Github Project](#)".

Reviewing Requests

If you have the review permission in a division and are logged in, you can click on the “Pending” link at the top of the screen to see a list of requests that are not in a final (paid or rejected) state, and are thus able to be reviewed. The number of requests that are in the “Evaluating” state is displayed in the number badge next to the “Pending” button.

In the list of requests, unevaluated requests have a yellow background, incomplete and rejected have a red background, approved (pending payout) have a blue one, and paid requests have a green background. To open a request, click the Request ID link (blue text).

EVE-SRP

Pending 2 Pay Outs Completed Submit Personal Admin Log Out

Filters

Request ID	Pilot	Ship	System	Status	Payout	Submit Time	Division
39861569	Paxswill	Guardian	WD-VTV	Evaluating	179,364,676.16	09 Jul 2014 @ 19:43	Test Alliance
39697412	Paxswill	Tristan	Hikkoken	Evaluating	9,158,708.44	09 Jul 2014 @ 09:12	Test Alliance

EVE-SRP v0.8.6-dev • [Github Project](#)

19 queries in 21.1 ms

In addition to the controls available to a normal user, reviewers have a few extra controls available. The base payout can be set by entering a value (in millions of ISK) and clicking the “Set” button.

EVE-SRP Pending **2** Pay Outs Completed Submit Personal Admin Log Out

Lossmail 39861569
Time of Loss 02 Jul 2014 19:26
Division Test Alliance • [Change Division](#)
Details
 I literally forgot how to broadcast for armor.
[Update Details](#)

Pilot Paxswill
Corporation Dreddit
Alliance Test Alliance Please Ignore
Location Catch / UX3-N2 / WD-VTV
Ship Guardian
Status Evaluating
Payout 179,364,676.16
User Notes [Add Note](#)

Set Base Payout 40 [Set](#)
Modifiers [Type](#)
Reason

[Comment](#)

EVE-SRP v0.8.6-dev • [Github Project](#)

To apply bonuses and/or deduction, enter an amount in the “Add Modifier” form, enter a reason for the modifier, and then select the type of modifier from the dropdown button labeled, “Type”. Absolute modifiers (adding or subtracting a set amount of ISK) are applied first, followed by percentage deductions/bonuses.

EVE-SRP Pending **2** Pay Outs Completed Submit Personal Admin Log Out

Lossmail 39861569
Time of Loss 02 Jul 2014 19:26
Division Test Alliance • [Change Division](#)
Details
 I literally forgot how to broadcast for armor.
[Update Details](#)

Pilot Paxswill
Corporation Dreddit
Alliance Test Alliance Please Ignore
Location Catch / UX3-N2 / WD-VTV
Ship Guardian
Status Evaluating
Payout 40000000.00
User Notes [Add Note](#)

Set Base Payout M ISK [Set](#)
Modifiers 15 [Type](#)
Reason You're awesome

[Comment](#)

EVE-SRP v0.8.6-dev • [Github Project](#)

Open # on this page in a new tab

EVE-SRP Pending **2** Pay Outs Completed Submit Personal Admin Log Out

Lossmail 39861569
Time of Loss 02 Jul 2014 19:26
Division Test Alliance • [Change Division](#)
Details
 I literally forgot how to broadcast for armor.
[Update Details](#)

Pilot Paxswill
Corporation Dreddit
Alliance Test Alliance Please Ignore
Location Catch / UX3-N2 / WD-VTV
Ship Guardian
Status Evaluating
Payout 46000000.00
User Notes [Add Note](#)

Set Base Payout M ISK [Set](#)

Modifiers [Type](#)

Reason

15.0% bonus [×](#)

You're awesome!

paxswill at 19:50 9 Jul 2014

[Comment](#)

EVE-SRP v0.8.6-dev • [Github Project](#)

If you make a mistake on a modifier and the request is still in the evaluating state, you can void the modifier by clicking the small “X”.

EVE-SRP Pending **2** Pay Outs Completed Submit Personal Admin Log Out

Lossmail 39861569
Time of Loss 02 Jul 2014 19:26
Division Test Alliance • [Change Division](#)
Details
 I literally forgot how to broadcast for armor.
[Update Details](#)

Pilot Paxswill
Corporation Dreddit
Alliance Test Alliance Please Ignore
Location Catch / UX3-N2 / WD-VTV
Ship Guardian
Status Evaluating
Payout 40000000.00
User Notes [Add Note](#)

Set Base Payout M ISK [Set](#)

Modifiers [Type](#)

Reason

15.0% bonus [×](#)

[Comment](#)

EVE-SRP v0.8.6-dev • [Github Project](#)

Once you have applied all the modifiers you want/need, you can change the status of the request using the same interface used for commenting. Enter a reason for the status change in the comment box, click the dropdown button to

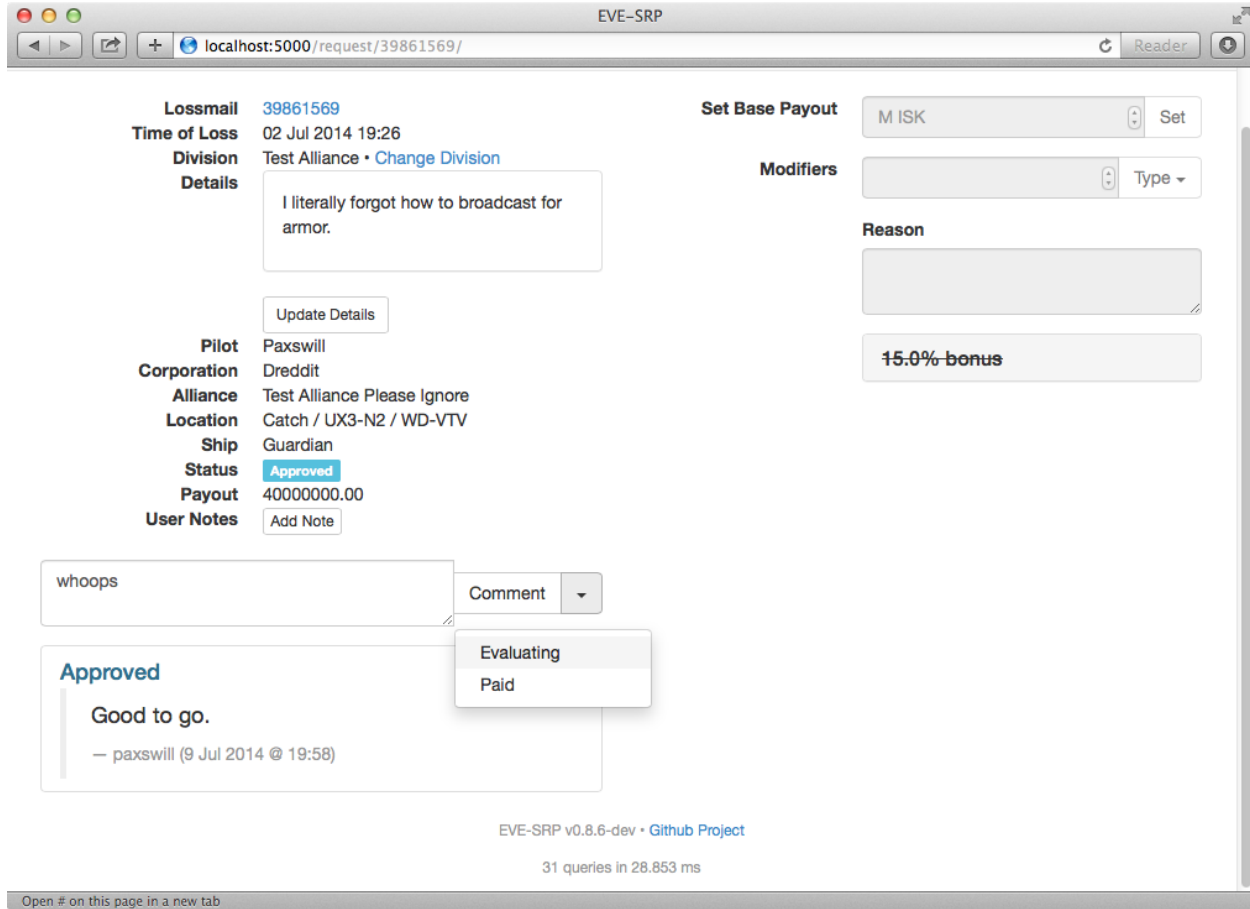
the right of the “Comment” button, and finally click the new status you want applied.

The screenshot shows the EVE-SRP web interface for request 39861569. The browser address bar shows 'localhost:5000/request/39861569/'. The page title is 'EVE-SRP'. The form contains the following fields and values:

- Lossmail:** 39861569
- Time of Loss:** 02 Jul 2014 19:26
- Division:** Test Alliance • [Change Division](#)
- Details:** I literally forgot how to broadcast for armor.
- Update Details:** [button]
- Pilot:** Paxswill
- Corporation:** Dreddit
- Alliance:** Test Alliance Please Ignore
- Location:** Catch / UX3-N2 / WD-VTV
- Ship:** Guardian
- Status:** Evaluating
- Payout:** 40000000.00
- User Notes:** Add Note [button]
- Set Base Payout:** M ISK [dropdown] Set [button]
- Modifiers:** [dropdown] Type [dropdown]
- Reason:** [text area]
- 15.0% bonus:** [button]
- Comment:** Good to go. [button]
- Comment Dropdown:** Incomplete, Evaluating, Rejected, Approved

At the bottom of the page, there is a link to the [Github Project](#) and a timestamp of 853 ms. A footer message says 'Open # on this page in a new tab'.

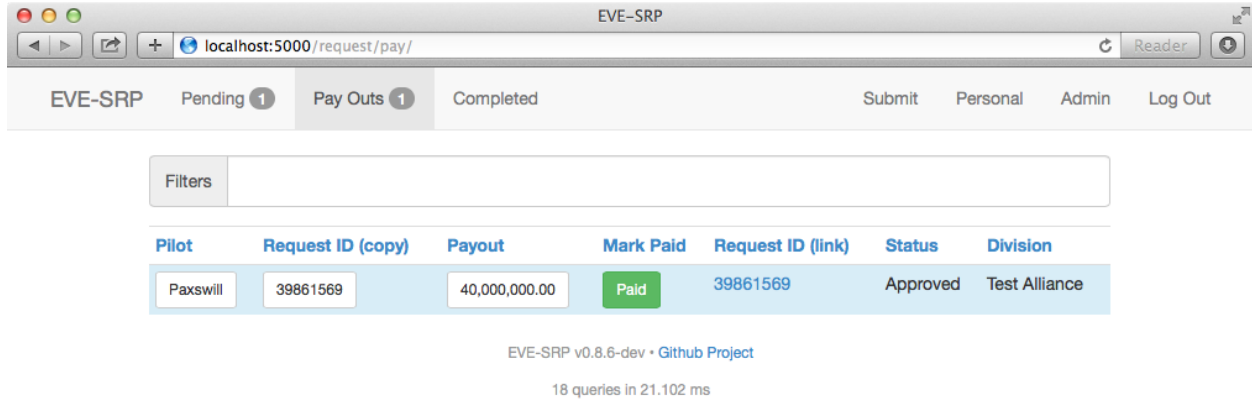
If you missed something and need to add or void a modifier, or even change the base payout, you can set approved (but not yet paid) requests back to evaluating.



Paying Out Requests

If you have the payer permission for a division, you can mark requests as paid. Typically this is handled by someone with access to the wallet in-game used to hold the SRP money.

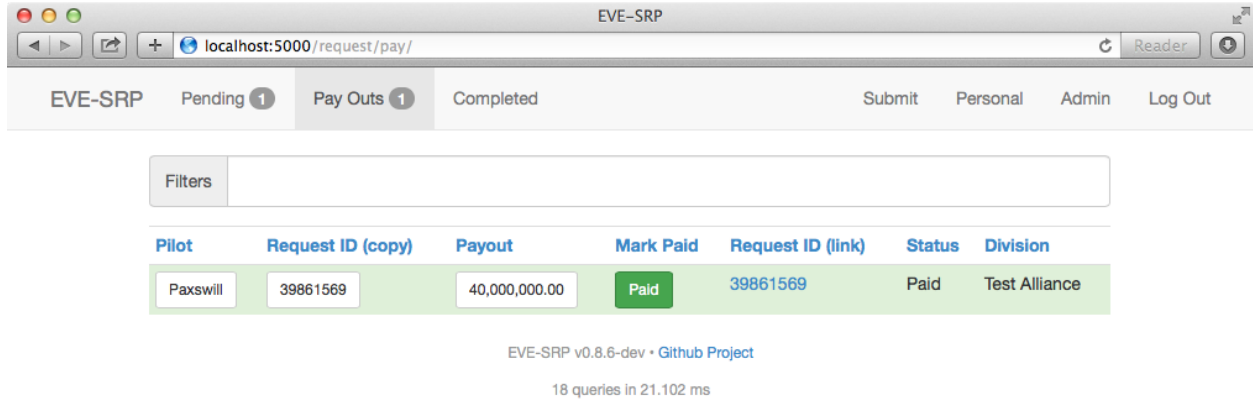
The number of requests pending payout is displayed in the number badge to the right of the “Pay Outs” button. This button is only visible if you have the payer permission. Click the button to see a list of approved requests.



This list tries to make paying out requests as quick as possible. Clicking one of the white buttons (under the “Pilot”, “Request ID (copy)”, or “Payout” columns) will copy the text within to your clipboard, making it quicker to enter the information in-game. The clipboard functionality requires Flash, so it should be done using an out of game browser. The work flow should be something like this:

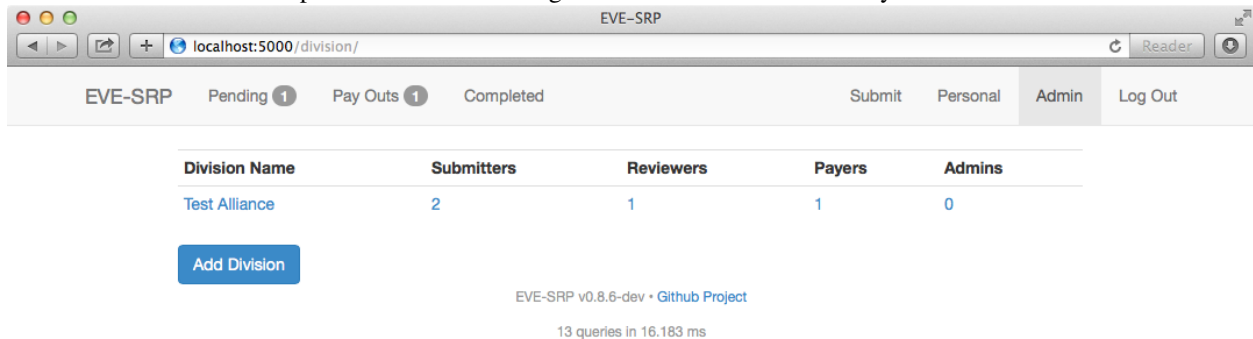
1. Copy Pilot name from app using standard web browser.
2. Paste the name in a search box for transferring money (either from a corp wallet or a personal wallet). Select the user and have the Give/Transfer ISK dialog box up.
3. Copy payout amount from app.
4. Paste payout amount into the amount box in-game.
5. Copy the request ID from the app.
6. Paste the request ID into the reason box in-game. Click the OK button to transfer the money.
7. Once the transfer has completed, click the green “Paid” button. This will mark the request as paid.

If you need to go back and fix something in a request, or to review them beforehand, you can click the request ID text (the blue link).



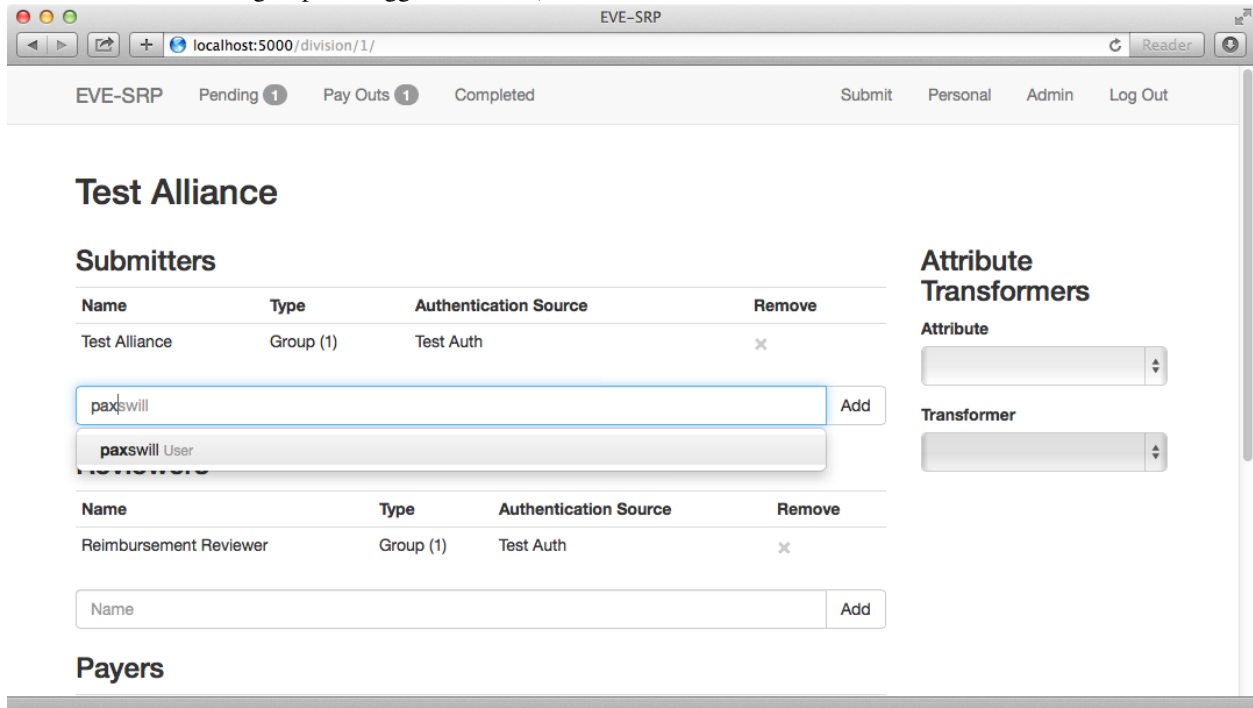
Administering Divisions

A fresh installation of EVE-SRP will not have any divisions configured, so one of the first actions after installation should be to configure divisions. If you have either the site administrator or administrator in a division, you will have an “Admin” button at the top of the screen. Clicking it will list all of the divisions you can administer.

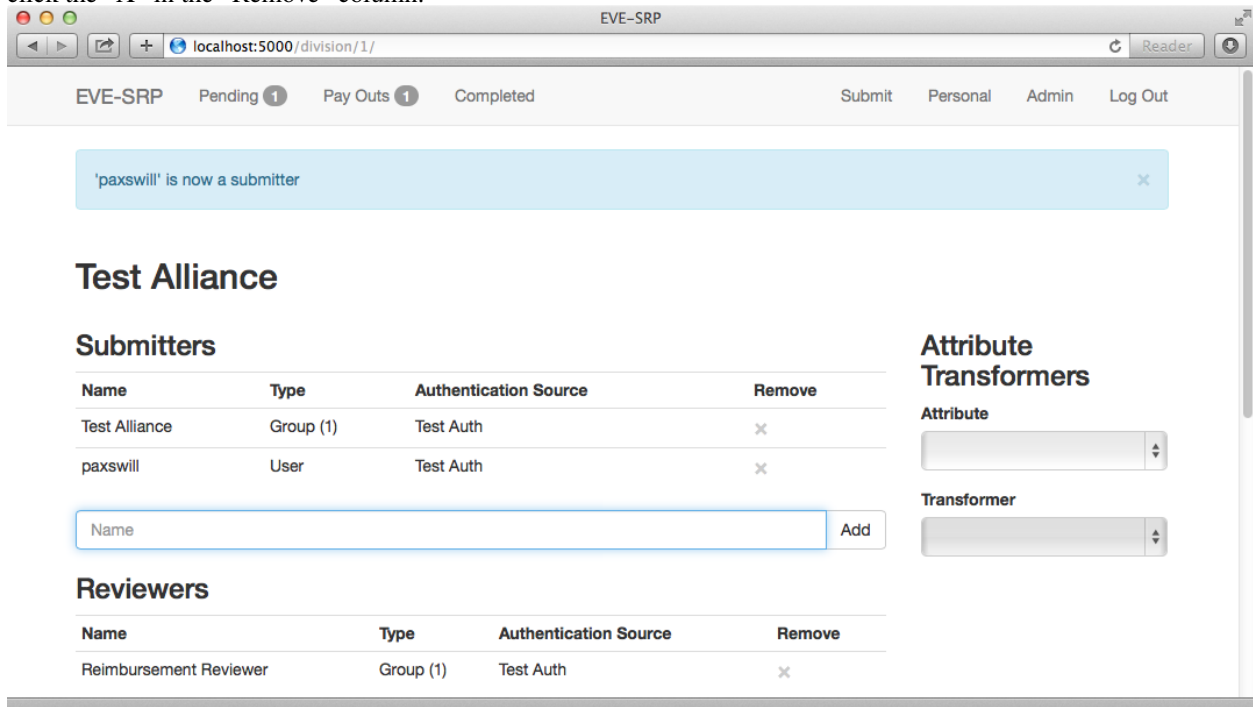


If you are a site administrator you will also see a button for creating divisions. To add a division, click the “Add Division” button, enter a name on the form, then click the “Create Division” button.

After creating a new division or clicking one of the links in the division listing, you will see the administration page for that division. To grant a permission to a user or group, start typing the name of that user or group in the text box corresponding to that permission. It will autocomplete their name if the app knows about it (i.e. if they've logged in before or a user in that group has logged in before).



Either click the correct entry, or finish typing it out and click the “Add” button. To revoke privileges to a user or group, click the “X” in the “Remove” column.



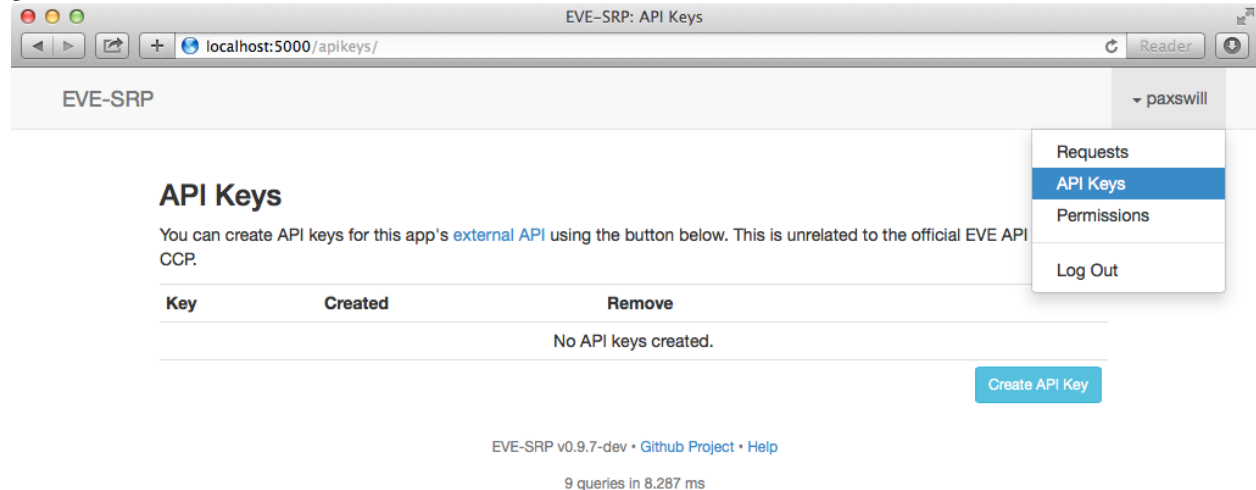
Divisions can be configured to have certain request attributes to be changed into links. This is covered in more detail in the (TODO) transformers section.

External API

EVE-SRP provides read-only access to requests you can access to external applications. Responses can be formatted as XML or JSON depending on your requirements. The URLs for the API are the same ones you access normally in a web browser, just in a different format.

API Keys

The first step to using the external API is to create an API key. Click the “Create API Key” button, and a key will be generated.



Open "http://localhost:5000/apikeys/" in a new tab

You can revoke API keys at any time by clicking the “X” in the “Remove” column. The key is the string of letters and numbers and can be copied to your clipboard by clicking on its button (requires Flash).

To use the API key, provide it as a parameter in the query string along with the desired format. The parameter name for the key is `apikey` and the field name for the format is `fmt`, and valid values are `json` or `xml`.

Lists of Requests

You can retrieve lists of up to 200 requests per page through the API. Filtering and sorting options are applied the same way they are when viewing the lists as HTML. In addition to the `personal`, `pending`, `pay` and `completed` lists exposed in the UI, there is an `all` route that will list all requests you have access to. As with the other lists that show requests other than your own, you must have a permission greater than ‘submitter’ granted to you in a division to access those lists.

JSON

In response to `http://example.com/request/personal/?apikey=dVbP0_SCPS12LnLpIZoJvemzeUU0OUeRT7noj`

```

{
  "api_keys": [
    {
      "id": 6,
      "key": "dVbP0_SCPS12LnLpIZoJvemzeUUOOUErT7nojbJW4_I,",
      "timestamp": "Thu, 10 Jul 2014 06:48:51 GMT"
    }
  ],
  "requests": [
    {
      "alliance": "Test Alliance Please Ignore",
      "base_payout": "40000000.00",
      "base_payout_str": "40,000,000.00",
      "corporation": "Dreddit",
      "details": "I literally forgot how to broadcast for armor.",
      "division": {
        "href": "/api/division/1/",
        "id": 1,
        "name": "Test Alliance"
      },
      "href": "/request/39861569/",
      "id": 39861569,
      "kill_timestamp": "Wed, 02 Jul 2014 19:26:00 GMT",
      "killmail_url": "https://zkillboard.com/kill/39861569/",
      "payout": "40000000.00",
      "payout_str": "40,000,000.00",
      "pilot": "Paxswill",
      "status": "paid",
      "submit_timestamp": "Wed, 09 Jul 2014 19:43:58 GMT",
      "submitter": {
        "href": "/api/user/1/",
        "id": 1,
        "name": "paxswill"
      }
    },
    {
      "alliance": "Test Alliance Please Ignore",
      "base_payout": "9158708.44",
      "base_payout_str": "9,158,708.44",
      "corporation": "Dreddit",
      "details": "crest mail?",
      "division": {
        "href": "/api/division/1/",
        "id": 1,
        "name": "Test Alliance"
      },
      "href": "/request/39697412/",
      "id": 39697412,
      "kill_timestamp": "Mon, 23 Jun 2014 16:06:00 GMT",
      "killmail_url": "https://zkillboard.com/kill/39697412/",
      "payout": "9158708.44",
      "payout_str": "9,158,708.44",
      "pilot": "Paxswill",
      "status": "paid",
      "submit_timestamp": "Wed, 09 Jul 2014 09:12:19 GMT",
      "submitter": {
        "href": "/api/user/1/",
        "id": 1,

```

```
    "name": "paxswill"
  }
}
]
```

XML

In response to `http://example.com/request/personal/?apikey=dVbP0_SCPS12LnLpIZoJvemzeUU00UErT7nojbJW4_I`

```
<?xml version="1.0" encoding="UTF-8" ?>
<response user="paxswill">
  <apikeyes>
    <apikey id="6">
      <key>dVbP0_SCPS12LnLpIZoJvemzeUU00UErT7nojbJW4_I,</key>
      <timestamp>2014-07-10T06:48:51.167054</timestamp>
    </apikey>
  </apikeyes>
  <requests>
    <request id="39861569" status="paid">
      <payout>
        <base pretty="40,000,000.00">40000000.00</base>
        <computed pretty="40,000,000.00">40000000.00</computed>
      </payout>
      <details>I literally forgot how to broadcast for armor.</details>
      <pilot>
        <alliance>Test Alliance Please Ignore</alliance>
        <corporation>Dreddit</corporation>
        <name>Paxswill</name>
      </pilot>
      <submit-timestamp>2014-07-09T19:43:58.126158</submit-timestamp>
      <kill-timestamp>2014-07-02T19:26:00</kill-timestamp>
      <division id="1" name="Test Alliance" />
      <submitter id="1" name="paxswill" />
      <killmail-url>https://zkillboard.com/kill/39861569/</killmail-url>
      <url>/request/39861569/</url>
      <ship>Guardian</ship>
      <location>
        <system>WD-VTV</system>
        <constellation>UX3-N2</constellation>
        <region>Catch</region>
      </location>
    </request>
    <request id="39697412" status="paid">
      <payout>
        <base pretty="9,158,708.44">9158708.44</base>
        <computed pretty="9,158,708.44">9158708.44</computed>
      </payout>
      <details>crest mail?</details>
      <pilot>
        <alliance>Test Alliance Please Ignore</alliance>
        <corporation>Dreddit</corporation>
        <name>Paxswill</name>
      </pilot>
      <submit-timestamp>2014-07-09T09:12:19.250893</submit-timestamp>
      <kill-timestamp>2014-06-23T16:06:00</kill-timestamp>
      <division id="1" name="Test Alliance" />
```

```

<submitter id="1" name="paxswill" />
<killmail-url>https://zkillboard.com/kill/39697412/</killmail-url>
<url>/request/39697412/</url>
<ship>Tristan</ship>
<location>
  <system>Hikkoken</system>
  <constellation>Ishaga</constellation>
  <region>Black Rise</region>
</location>
</request>
</requests>
</response>

```

RSS

An RSS feed for requests in a list is available by adding `/rss.xml` to the end of a list URL. For example, the URL for the feed of pending requests would be `http://example.com/request/pending/rss.xml?apikey=dVbP0_SCPS12LnLpIZoJvemzeUUOOUErT7nojbJW4_I, &fr`

Request Details

If you need details beyond that provided in the lists of requests, or to look up information on a specific request you can access a request's URL through the API. For example, the request for killmail [#39861569](#) in JSON format could be retrieved with the URL `http://example.com/request/39861569/?apikey=dVbP0_SCPS12LnLpIZoJvemzeUUOOUErT7nojbJW4_I, &fr`. The path for an individual requests is also returned as part of the response in request listings.

JSON

```

{
  "actions": [
    {
      "id": 2,
      "note": "",
      "timestamp": "Thu, 10 Jul 2014 06:37:09 GMT",
      "type": "paid",
      "user": {
        "href": "/api/user/1/",
        "id": 1,
        "name": "paxswill"
      }
    },
    {
      "id": 1,
      "note": "Good to go.",
      "timestamp": "Wed, 09 Jul 2014 19:58:56 GMT",
      "type": "approved",
      "user": {
        "href": "/api/user/1/",
        "id": 1,
        "name": "paxswill"
      }
    }
  ],
  ]

```

```
"alliance": "Test Alliance Please Ignore",
"base_payout": "40000000.00",
"base_payout_str": "40,000,000.00",
"corporation": "Dreddit",
"current_user": {
  "href": "/api/user/1/",
  "id": 1,
  "name": "paxswill"
},
"details": "I literally forgot how to broadcast for armor.",
"division": {
  "href": "/api/division/1/",
  "id": 1,
  "name": "Test Alliance"
},
"href": "/request/39861569/",
"id": 39861569,
"kill_timestamp": "Wed, 02 Jul 2014 19:26:00 GMT",
"killmail_url": "https://zkillboard.com/kill/39861569/",
"modifiers": [
  {
    "id": 1,
    "note": "You're awesome!",
    "timestamp": "Wed, 09 Jul 2014 19:50:10 GMT",
    "user": {
      "href": "/api/user/1/",
      "id": 1,
      "name": "paxswill"
    },
    "value": 0.15,
    "value_str": "15.0% bonus",
    "void": {
      "timestamp": "Wed, 09 Jul 2014 19:58:00 GMT",
      "user": {
        "href": "/api/user/1/",
        "id": 1,
        "name": "paxswill"
      }
    }
  }
],
"payout": "40000000.00",
"payout_str": "40,000,000.00",
"pilot": "Paxswill",
"status": "paid",
"submit_timestamp": "Wed, 09 Jul 2014 19:43:58 GMT",
"submitter": {
  "href": "/api/user/1/",
  "id": 1,
  "name": "paxswill"
},
"valid_actions": [
  "approved",
  "evaluating"
]
}
```

XML

```

<?xml version="1.0" encoding="UTF-8" ?>
<response user="paxswill">
  <request id="39861569" status="paid">
    <payout>
      <base pretty="40,000,000.00">40000000.00</base>
      <computed pretty="40,000,000.00">40000000.00</computed>
    </payout>
    <details>I literally forgot how to broadcast for armor.</details>
    <pilot>
      <alliance>Test Alliance Please Ignore</alliance>
      <corporation>Dreddit</corporation>
      <name>Paxswill</name>
    </pilot>
    <submit-timestamp>2014-07-09T19:43:58.126158</submit-timestamp>
    <kill-timestamp>2014-07-02T19:26:00</kill-timestamp>
    <division id="1" name="Test Alliance" />
    <submitter id="1" name="paxswill" />
    <killmail-url>https://zkillboard.com/kill/39861569/</killmail-url>
    <url>/request/39861569/</url>
    <ship>Guardian</ship>
    <location>
      <system>WD-VTV</system>
      <constellation>UX3-N2</constellation>
      <region>Catch</region>
    </location>
    <actions>
      <action id="2" type="paid">
        <note></note>
        <timestamp>2014-07-10T06:37:09.242568</timestamp>
        <user id="1" name="paxswill" />
      </action>
      <action id="1" type="approved">
        <note>Good to go.</note>
        <timestamp>2014-07-09T19:58:56.524278</timestamp>
        <user id="1" name="paxswill" />
      </action>
    </actions>
    <modifiers>
      <modifier id="1">
        <note>You&#39;re awesome!</note>
        <user id="1" name="paxswill" />
        <value>15.0% bonus</value>
        <timestamp>2014-07-09T19:50:10.909394</timestamp>
        <void id="1" name="paxswill">
          <timestamp>2014-07-09T19:58:00.069323</timestamp>
        </void>
      </modifier>
    </modifiers>
  </request>
</response>

```

Developers Guide

Authentication

Authentication in EVE-SRP was designed from the start to allow for multiple different authentication systems and to make it easy to integrate it with an existing authentication system.

As an exercise in how to write your own authentication plugin, let's write one that doesn't rely on an external service. We'll need to subclass two classes for this; `AuthMethod` and `User`

Let's start with subclassing `User`. This class is mapped to an SQL table using SQLAlchemy's declarative extension (more specifically, the Flask-SQLAlchemy plugin for Flask). The parent class automatically sets up the table name and inheritance mapper arguments for you, so all you need to do is provide the `id` attribute that links your class with the parent class and an attribute to store the password hash. In the example below, we're using the `simple-pbkdf2` package to provide the password hashing. We also have a checking method to make life easier for us later.

```
import os
from hashlib import sha512
from evesrp import db
from evesrp.auth.models import User
from pbkdf2 import pbkdf2_bin

class LocalUser(User):
    id = db.Column(db.Integer, db.ForeignKey(User.id), primary_key=True)
    password = db.Column(db.LargeBinary(24), nullable=False)
    salt = db.Column(db.LargeBinary(24), nullable=False)

    def __init__(self, name, password, authmethod, **kwargs):
        self.salt = os.urandom(24)
        self.password = pbkdf2_bin(password.encode('utf-8'), self.salt,
                                    iterations=10000)
        super(LocalUser, self).__init__(name, authmethod, **kwargs)

    def check_password(self, password):
        key = pbkdf2_bin(password.encode('utf-8'), self.salt,
                          iterations=10000)
        matched = 0
        for a, b in zip(self.password, key):
            matched |= ord(a) ^ ord(b)
        return matched == 0
```

`AuthMethod` subclasses have four methods they can implement to customize thier behavior.

- `AuthMethod.form()` returns a `Form` subclass that represents the necessary fields.

- `AuthMethod.login()` performs the actual login process. As part of this, it is passed an instance of the class given by `AuthMethod.form()` with the submitted data via the `form` argument.
- For those authentication methods that requires a secondary view/route, the `AuthMethod.view()` method can be implemented to handle requests made to `login/safe_name` where `safe_name` is the output of `AuthMethod.safe_name`.
- Finally, the initializer should be overridden to provide a default name for your `AuthMethod` other than `Base Authentication`.
- Finally, the initializer can be overridden to handle specialized configurations.

With these in mind, let's implement our `AuthMethod` subclass:

```
from evesrp.auth import AuthMethod
from flask import redirect, url_for, render_template, request
from flask_wtf import Form
from sqlalchemy.orm.exc import NoResultFound
from wtforms.fields import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, EqualTo

class LocalLoginForm(Form):
    username = StringField('Username', validators=[InputRequired()])
    password = PasswordField('Password', validators=[InputRequired()])
    submit = SubmitField('Log In')

class LocalCreateUserForm(Form):
    username = StringField('Username', validators=[InputRequired()])
    password = PasswordField('Password', validators=[InputRequired(),
        EqualTo('password_repeat', message='Passwords must match')])
    password_repeat = PasswordField(
        'Repeat Password', validators=[InputRequired()])
    submit = SubmitField('Log In')

class LocalAuth(AuthMethod):

    def form(self):
        return LocalLoginForm

    def login(self, form):
        # form has already been validated, we just need to process it.
        try:
            user = LocalUser.query.filter_by(name=form.username.data).one()
        except NoResultFound:
            flash("No user found with that username.", 'error')
            return redirect(url_for('login.login'))
        if user.check_password(form.password.data):
            self.login_user(user)
            return redirect(request.args.get('next') or url_for('index'))
        else:
            flash("Incorrect password.", 'error')
            return redirect(url_for('login.login'))

    def view(self):
        form = LocalCreateUserForm()
        if form.validate_on_submit():
            user = LocalUser(form.username.data, form.password.data)
```

```

        db.session.add(user)
        db.session.commit()
        self.login_user(user)
        return redirect(url_for('index'))
# form.html is a template included in Eve-SRP that renders all
# elements of a form.
        return render_template('form.html', form=form)

```

That's all that's necessary for a very simple `AuthMethod`. This example cuts some corners, and isn't ready for production-level use, but it serves as a quick example of what's necessary to write a custom authentication method. Feel free to look at the sources for the included `AuthMethods` below to gather ideas on how to use more complicated mechanisms.

Included Authentication Methods

Brave Core

```

class evesrp.auth.bravecore.BraveCore(client_key, server_key, identifier,
                                     url='https://core.braveineve.com', **kwargs)
    Bases: evesrp.auth.AuthMethod

```

```

__init__(client_key, server_key, identifier, url='https://core.braveineve.com', **kwargs)
    Authentication method using a Brave Core instance.

```

Uses the native Core API to authenticate users. Currently only supports a single character at a time due to limitations in Core's API.

Parameters

- **client_key** (*str*) – The client's private key in hex form.
- **server_key** (*str*) – The server's public key for this app in hex form.
- **identifier** (*str*) – The identifier for this app in Core.
- **url** (*str*) – The URL of the Core instance to authenticate against. Default: 'https://core.braveineve.com'
- **name** (*str*) – The user-facing name for this authentication method. Default: 'Brave Core'

TEST Legacy

```

class evesrp.auth.testauth.TestAuth(api_key=None, **kwargs)
    Bases: evesrp.auth.AuthMethod

```

```

__init__(api_key=None, **kwargs)
    Authentication method using TEST Auth's legacy (a.k.a v1) API.

```

Parameters

- **api_key** (*str*) – (optional) An Auth API key. Without this, only primary characters are able to be accessed/used.
- **name** (*str*) – The user-facing name for this authentication method. Default: 'Test Auth'

OAuth

A number of external authentication services have an OAuth provider for external applications to use with their API. To facilitate usage of these services, an `OAuthMethod` class has been provided for easy integration. Subclasses will need to implement the `get_user()`, `get_pilots()` and `get_groups()` methods. Additionally, implementations for JFLP's `provider` and TEST's `provider` have been provided as a reference.

```
class evesrp.auth.oauth.OAuthMethod(**kwargs)
```

```
    __init__(**kwargs)
```

Abstract `AuthMethod` for OAuth-based login methods.

Implementing classes need to implement `get_user()`, `get_pilots()`, and `get_groups()`.

In addition to the keyword arguments from `AuthMethod`, this initializer accepts the following arguments that will be used in the creation of the `OAuthMethod.oauth` object (See the documentation for `OAuthRemoteApp` for more details):

- `client_id`
- `client_secret`
- `scope`
- `access_token_url`
- `refresh_token_url`
- `authorize_url`
- `access_token_params`
- `method`

As a convenience, the `key` and `secret` keyword arguments will be treated as `consumer_key` and `consumer_secret` respectively. The `name` argument is used for both `AuthMethod` and for `OAuthRemoteApp`.

Subclasses for providers that may be used by more than one entity are encouraged to provide their own defaults for the above arguments.

The redirect URL for derived classes is based off of the `safe_name` of the implementing `AuthMethod`, specifically the URL for `view()`. For example, the default redirect URL for `TestOAuth` is similar to `https://example.com/login/test_oauth/` (Note the trailing slash, it is significant).

Parameters `default_token_expiry` (`int`) – The default time (in seconds) access tokens are valid for. Defaults to 5 minutes.

```
get_groups()
```

Returns a `list` of `Groups` for the given token.

Like `get_user()` and `get_pilots()`, this method is to be implemented by `OAuthMethod` subclasses to return a `list` of `Groups` associated with the account for the given access token.

Return type `list` of `Groups`.

```
get_pilots()
```

Return a `list` of `Pilots` for the given token.

Like `get_user()`, this method is to be implemented by `OAuthMethod` subclasses to return a `list` of `Pilots` associated with the account for the given access token.

Return type `list` of `Pilots`.

get_user()

Returns the `OAuthUser` instance for the current token.

This method is to be implemented by subclasses of `OAuthMethod` to use whatever APIs they have access to to get the user account given an access token.

Return type `OAuthUser`

is_admin(user)

Returns whether this user should be treated as a site-wide administrator.

The default implementation checks if the user's name is contained within the list of administrators supplied as an argument to `OAuthMethod`.

Parameters `user` (`OAuthUser`) – The user to check.

Return type `bool`

refresh(user)

Refreshes the current user's information.

Attempts to refresh the pilots and groups for the given user. If the current access token has expired, the refresh token is used to get a new access token.

view()

Handle creating and/or logging in the user and updating their `Pilots` and `Groups`.

EVE SSO

```
class evesrp.auth.evesso.EveSSO(singularity=False, **kwargs)
```

Bases: `evesrp.auth.oauth.OAuthMethod`

get_groups()

Set the user's groups for their pilot.

At this time, Eve SSO only gives us character access, so they're just set to the pilot's corporation, and if they have on their alliance as well. In the future, this method may also add groups for mailing lists.

J4OAuth

```
class evesrp.auth.j4oauth.J4OAuth(base_url='https://j4lp.com/oauth/api/v1/', **kwargs)
```

Bases: `evesrp.auth.oauth.OAuthMethod`

```
__init__(base_url='https://j4lp.com/oauth/api/v1/', **kwargs)
```

`AuthMethod` for using `J4OAuth` as an authentication source.

Parameters

- **authorize_url** (*str*) – The URL to request OAuth authorization tokens. Default: `'https://j4lp.com/oauth/authorize'`.
- **access_token_url** (*str*) – The URL for OAuth token exchange. Default: `'https://j4lp.com/oauth/token'`.
- **base_str** (*str*) – The base URL for API requests. Default: `'https://j4lp.com/oauth/api/v1/'`.
- **request_token_params** (*dict*) – Additional parameters to include with the authorization token request. Default: `{'scope': ['auth_info', 'auth_groups', 'characters']}`.

- **access_token_method** (*str*) – HTTP Method to use for exchanging authorization tokens for access tokens. Default: 'GET'.
- **name** (*str*) – The name for this authentication method. Default: 'J4OAuth'.

TestOAuth

class `evesrp.auth.testoauth.TestOAuth` (*devtest=False, **kwargs*)

Bases: `evesrp.auth.oauth.OAuthMethod`

__init__ (*devtest=False, **kwargs*)

`AuthMethod` using TEST Auth's OAuth-based API for authentication and authorization.

Parameters

- **admins** (*list*) – Two types of values are accepted as values in this list, either a string specifying a user's primary character's name, or their Auth ID as an integer.
- **devtest** (*bool*) – Testing parameter that changes the default domain for URLs from 'https://auth.pleaseignore.com' to 'https://auth.devtest.pleaseignore.com'. Default: False.
- **authorize_url** (*str*) – The URL to request OAuth authorization tokens. Default: 'https://auth.pleaseignore.com/oauth2/authorize'.
- **access_token_url** (*str*) – The URL for OAuth token exchange. Default: 'https://auth.pleaseignore.com/oauth2/access_token'.
- **base_str** (*str*) – The base URL for API requests. Default: 'https://auth.pleaseignore.com/api/v3/'.
- **request_token_params** (*dict*) – Additional parameters to include with the authorization token request. Default: {'scope': 'private-read'}.
- **access_token_method** (*str*) – HTTP Method to use for exchanging authorization tokens for access tokens. Default: 'POST'.
- **name** (*str*) – The name for this authentication method. Default: 'Test OAuth'.

Low-Level API

class `evesrp.auth.PermissionType`

Enumerated type for the types of permissions available.

elevated

Returns a `frozenset` of the permissions above `submit`.

all

Returns a `frozenset` of all possible permission values.

admin = <admin>

`Division`-level administrator permission

audit = <audit>

A special permission for allowing read-only elevated access

pay = <pay>

Permission for payers in a `Division`.

review = <review>

Permission for reviewers of requests in a `Division`.

submit = <submit>

Permission allowing submission of `Requests` to a `Division`.

class `evesrp.auth.AuthMethod` (*admins=None*, *name='Base Authentication'*, ***kwargs*)

Represents an authentication mechanism for users.

__init__ (*admins=None*, *name='Base Authentication'*, ***kwargs*)

Parameters

- **admins** (*list*) – A list of usernames to treat as site-wide administrators. Useful for initial setup.
- **name** (*str*) – The user-facing name for this authentication method.

form ()

Return a `flask_wtf.Form` subclass to login with.

login (*form*)

Process a validated login form.

You must return a valid response object.

static login_user (*user*)

Signal to the authentication systems that a new user has logged in.

Handles calling `flask_login.login_user()` and any other related housekeeping functions for you.

Parameters *user* (`User`) – The user that has been authenticated and is logging in.

refresh (*user*)

Refresh a user's information (if possible).

The `AuthMethod` should attempt to refresh the given user's information as if they were logging in for the first time.

Parameters *user* (`User`) – The user to refresh.

Returns Whether or not the refresh attempt succeeded.

Return type `bool`

safe_name

Normalizes a string to be a valid Python identifier (along with a few other things).

Specifically, all letters are lower cased and non-ASCII and whitespace are replaced by underscores.

Returns The normalized string.

Rtype `str`

view ()

Optional method for providing secondary views.

`evesrp.views.login.auth_method_login()` is configured to allow both GET and POST requests, and will call this method as soon as it is known which auth method is meant to be called. The path for this view is `/login/self.safe_name/`, and can be generated with `url_for('login.auth_method_login', auth_method=self.safe_name)`.

The default implementation redirects to the main login view.

class `evesrp.auth.models.Entity` (*name*, *authmethod*, ***kwargs*)

Private class for shared functionality between `User` and `Group`.

This class defines a number of helper methods used indirectly by `User` and `Group` subclasses such as automatically defining the table name and mapper arguments.

This class should *not* be inherited from directly, instead either `User` or `Group` should be used.

authmethod

The name of the `AuthMethod` for this entity.

entity_permissions

`Permissions` associated specifically with this entity.

has_permission (*permissions*, *division_or_request=None*)

Returns if this entity has been granted a permission in a division.

If *division_or_request* is `None`, this method checks if this group has the given permission in *any* division.

Parameters

- **permissions** (*iterable*) – The series of permissions to check
- **division_or_request** – The division to check. May also be `None` or an SRP request.

Return type `bool`

name

The name of the entity. Usually a nickname.

class `evesrp.auth.models.User` (*name*, *authmethod*, ***kwargs*)

Bases: `evesrp.auth.models.Entity`

User base class.

Represents users who can submit, review and/or pay out requests. It also supplies a number of convenience methods for subclasses.

actions

`Actions` this user has performed on requests.

admin

If the user is an administrator. This allows the user to create and administer divisions.

get_id()

Part of the interface for Flask-Login.

groups

`Groups` this user is a member of

is_active

Part of the interface for Flask-Login.

is_anonymous

Part of the interface for Flask-Login.

is_authenticated

Part of the interface for Flask-Login.

pilots

`Pilots` associated with this user.

requests

`Requests` this user has submitted.

submit_divisions()

Get a list of the divisions this user is able to submit requests to.

Returns A list of tuples. The tuples are in the form (division.id, division.name)

Return type `list`

```

class evesrp.auth.models.Pilot(user, name, id_)
    Represents an in-game character.

    __init__(user, name, id_)
        Create a new Pilot instance.

        Parameters
        • user (User) – The user this character belongs to.
        • name (str) – The name of this character.
        • id (int) – The CCP-given characterID number.

    name
        The name of the character

    requests
        The Requests filed with lossmails from this character.

    user
        The User this character belongs to.
class evesrp.auth.models.APIKey(user)
    Represents an API key for use with the External API.

    hex_key
        The key data in a modified base-64 format safe for use in URLs.

    key
        The raw key data.

    user
        The User this key belongs to.
class evesrp.auth.models.Note(user, noter, note)
    A note about a particular User.

    content
        The actual contents of this note.

    noter
        The author of this note.

    user
        The User this note refers to.
class evesrp.auth.models.Group(name, authmethod, **kwargs)
    Bases: evesrp.auth.models.Entity
    Base class for a group of users.

    Represents a group of users. Usable for granting permissions to submit, evaluate and pay.

    permissions
        Synonym for entity_permissions

    users
        User s that belong to this group.
class evesrp.auth.models.Permission(division, permission, entity)

    __init__(division, permission, entity)
        Create a Permission object granting an entity access to a division.

```

division

The division this permission is granting access to

entity

The `Entity` being granted access

permission

The permission being granted.

class `evesrp.auth.models.Division(name)`

A reimbursement division.

A division has (possibly non-intersecting) groups of people that can submit requests, review requests, and pay out requests.

division_permissions

All `Permissions` associated with this division.

name

The name of this division.

permissions

The permissions objects for this division, mapped via their permission names.

requests

Request s filed under this division.

transformers

A mapping of attribute names to `Transformer` instances.

class `evesrp.auth.models.TransformerRef(**kwargs)`

Stores associations between `Transformers` and `Divisions`.

attribute_name

The attribute this transformer is applied to.

division

The division the transformer is associated with

transformer

The transformer instance.

Killmail Handling

EVE-SRP relies on outside sources for its killmail information. Whether that source is ESI, zKillboard, or some private killboard does not matter, there just has to be some sort of access to the information.

The interface for `Killmail` is fairly simple. It provides a number of attributes, and for those that correspond to in-game entities, it also provides their ID number. The default implementation has all values set to `None`. If a killmail is invalid in some way, it can be signaled either by raising a `ValueError` or `LookupError` in the killmail's `__init__()` method or by defining a `Killmail.verified` property and returning `False` from it when the killmail is invalid.

Two implementations for creating a `Killmail` from a URL are included: `ESIMail` is created from a ESI external killmail link, and `ZKillmail` is created from a zKillboard details link.

Extension Examples

The reasoning behind having killmails handled in a separate class was for administrators to be able to customize behavior. Here're a few useful snippets that may be useful for your situation.

Restricting Valid zKillboards

ZKillmail by default will accept any link that looks and acts like a zKillboard instance. It does *not* restrict itself to any particular domain name, but it makes allowances for this common requirement.

```
from evesrp.killmail import ZKillmail

class OnlyZKillboard(ZKillmail):
    def __init__(self, *args, **kwargs):
        super(OnlyZKillboard, self).__init__(*args, **kwargs)
        if self.domain != 'zkillboard.com':
            raise ValueError(u"This killmail is from the wrong killboard.")
```

Submitting ESI Links to zKillboard

To streamline the process for users, you can accept ESI killmail links and then submits them to zKillboard.com and uses the new zKillboard.com link as the canonical URL for the request.

```
from decimal import Decimal
from flask import Markup
from evesrp.killmail import ESIMail

class SubmittedESIZKillmail(ESIMail):
    """Accepts and validates ESI killmail links, but submits them to
    ZKillboard and substitutes the zKB link in as the canonical link
    """

    def __init__(self, url, **kwargs):
        # Let ESIMail validate the ESI link
        super(SubmittedESIZKillmail, self).__init__(url, **kwargs)
        # Submit the ESI URL to ZKillboard
        resp = self.requests_session.post('https://zkillboard.com/post/',
                                          data={'killmailurl': url})
        # Use the URL we get from ZKillboard as the new URL (if it's successful).
        if self.kill_id in resp.url:
            self.url = resp.url
        else:
            # Leave the ESI URL as-is and finish
            return
        # Grab zkb's data from their API
        api_url = ('https://zkillboard.com/api/no-attackers/'
                  'no-items/killID/{}'.format(self.kill_id))
        zkb_api = self.requests_session.get(api_url)
        retrieval_error = LookupError(u"Error retrieving killmail data (zKB): {}".format(zkb_api.status_code))
        if zkb_api.status_code != 200:
            raise retrieval_error
        try:
            json = zkb_api.json()
```

```
except ValueError as e:
    raise retrieval_error
try:
    json = json[0]
except IndexError as e:
    raise LookupError(u"Invalid killmail: {}".format(url))
# Recent versions of zKillboard calculate a loss' value.
try:
    self.value = Decimal(json[u'zkb'][u'totalValue'])
except KeyError:
    self.value = Decimal(0)

description = Markup(u'An ESI external killmail link that will be '
                    u'automatically submitted to <a href="https://'+
                    u'zkillboard.com">zKillboard.com</a>.'.)
```

Setting Base Payouts from a Spreadsheet

If you have standardized payout values in a Google spreadsheet, you can set `Request.base_payout` to the values in this spreadsheet. This is assuming your spreadsheet is set up with ship hull names in one column and payouts in another column. Both Columns need to have a header ('Hull' and 'Payout' in the example below). This uses the Google Data Python Client which only supports Python 2, and can be installed with `pip install gdata`.

```
import gdata.spreadsheets.client
from decimal import Decimal

# patch the spreadsheet's client to use the public feeds
gdata.spreadsheets.client.PRIVATE_WORKSHEETS_URL = \
    gdata.spreadsheets.client.WORKSHEETS_URL
gdata.spreadsheets.client.WORKSHEETS_URL = ('https://spreadsheets.google.com/'
                                             'feeds/worksheets/%s/public/full')
gdata.spreadsheets.client.PRIVATE_LISTS_URL = \
    gdata.spreadsheets.client.LISTS_URL
gdata.spreadsheets.client.LISTS_URL = ('https://spreadsheets.google.com/feeds/'
                                         'list/%s/%s/public/full')

class SpreadsheetPayout(ZKillmail):

    # The spreadsheet's key
    # (https://docs.google.com/spreadsheets/d/THE_PART_HERE/edit).
    # Make sure the spreadsheet has been published (File->Publish to web...)
    spreadsheet_key = 'THE_PART_HERE'

    # The name of the worksheet with the payouts
    worksheet_name = 'Payouts'

    # The header for the hull column (always lowercase, the Google API
    # lowercases it).
    hull_key = 'hull'

    # And the same for the payout column
    payout_key = 'payout'

    client = gdata.spreadsheets.client.SpreadsheetsClient()
```

```

@property
def value(self):
    # Find the worksheet
    sheets = self.client.get_worksheets(self.spreadsheet_key)
    for sheet in sheets.entry:
        if sheet.title.text == self.worksheet_name:
            worksheet_id = sheet.get_worksheet_id()
            break
    else:
        return Decimal('0')
    # Read the worksheet's data
    lists = self.client.get_list_feed(self.spreadsheet_key, worksheet_id,
        query=gdata.spreadsheets.client.ListQuery(sq='{}={}'.format(
            self.hull_key, self.ship)))
    for entry in lists.entry:
        return Decimal(entry.get_value(self.payout_key))
    return Decimal('0')

```

Developer API

class `evesrp.killmail.Killmail` (***kwargs*)

Base killmail representation.

kill_id

The ID integer of this killmail. Used by most killboards and by CCP to refer to killmails.

ship_id

The typeID integer of for the ship lost for this killmail.

ship

The human readable name of the ship lost for this killmail.

pilot_id

The ID number of the pilot who lost the ship. Referred to by CCP as `characterID`.

pilot

The name of the pilot who lost the ship.

corp_id

The ID number of the corporation `pilot` belonged to at the time this kill happened.

corp

The name of the corporation referred to by `corp_id`.

alliance_id

The ID number of the alliance `corp` belonged to at the time of this kill, or `None` if the corporation wasn't in an alliance at the time.

alliance

The name of the alliance referred to by `alliance_id`.

url

A URL for viewing this killmail's information later. Typically an online killboard such as [zKillboard](#), but other kinds of links may be used.

value

The estimated ISK loss for the ship destroyed in this killmail. This is an optional attribute, and is `None` if unsupported. If this attribute is set, it should be a `Decimal` or at least a type that can be used as the value for the `Decimal` constructor.

timestamp

The date and time that this kill occurred as a `datetime.datetime` object (with a UTC timezone).

verified

Whether or not this killmail has been API verified (or more accurately, if it is to be trusted when making a `Request`).

system

The name of the system where the kill occurred.

system_id

The ID of the system where the kill occurred.

constellation

The name of the constellation where the kill occurred.

region

The name of the region where the kill occurred.

__init__ (***kwargs*)

Initialize a `Killmail` with `None` for all attributes.

All subclasses of this class, (and all mixins designed to be used with it) must call `super().__init__(**kwargs)` to ensure all initialization is done.

Param keyword arguments corresponding to attributes.

__iter__ ()

Iterate over the attributes of this killmail.

Yields tuples in the form (`'<name>'`, `<value>`). This is used by `Request.__init__` to initialize its data quickly. `<name>` in the returned tuples is the name of the attribute on the `Request`.

description = I'A generic Killmail. If you see this text, you need to configure your application.'

A user-facing description of what kind of killmails this `Killmail` validates/handles. This text is displayed below the text field for a killmail URL to let users know what kinds of links are acceptable.

```
class evesrp.killmail.ZKillmail(url, **kwargs)
```

Bases: `evesrp.killmail.ESIMail`

domain

The domain name of this killboard.

```
class evesrp.killmail.ESIMail(url, **kwargs)
```

Bases: `evesrp.killmail.Killmail`, `evesrp.killmail.RequestsSessionMixin`, `evesrp.killmail.LocationMixin`

A killmail with data sourced from a ESI killmail link.

__init__ (*url*, ***kwargs*)

Create a killmail from a ESI killmail link.

Parameters `url` (*str*) – the ESI killmail URL.

Raises

- **ValueError** – if `url` is not a ESI URL.
- **LookupError** – if the ESI API response is in an unexpected format.

```
class evesrp.killmail.RequestsSessionMixin(requests_session=None, **kwargs)
```

Mixin for providing a `requests.Session`.

The shared session allows HTTP user agents to be set properly, and for possible connection pooling.

requests_session

A [Session](#) for making HTTP requests.

__init__ (*requests_session=None, **kwargs*)

Set up a [Session](#) for making HTTP requests.

If an existing session is not provided, one will be created.

Parameters **requests_session** – an existing session to use.

class `evesrp.killmail.ShipNameMixin`

Killmail mixin providing `Killmail.ship` from `Killmail.ship_id`.

ship

Looks up the ship name using `Killmail.ship_id`.

class `evesrp.killmail.LocationMixin`

Killmail mixin for providing solar system, constellation and region names from `Killmail.system_id`.

constellation

Provides the constellation name using `Killmail.system_id`.

region

Provides the region name using `Killmail.system_id`.

system

Provides the solar system name using `Killmail.system_id`.

Views

`evesrp.views.index()`

The index page for EVE-SRP.

Login

`evesrp.views.login.auth_method_login(auth_method)`

Trampoline for `AuthMethod`-specific views.

See `Authmethod.view` for more details.

`evesrp.views.login.login()`

Presents the login form and processes responses from that form.

When a POST request is recieved, this function passes control to the appropriate `login` method.

`evesrp.views.login.login_loader(userid)`

Pull a user object from the database.

This is used for loading users from existing sessions.

`evesrp.views.login.logout()`

Logs the current user out.

Redirects to `index()`.

Divisions

`evesrp.views.divisions.add_division()`

Present a form for adding a division and also process that form.

Only accessible to administrators.

`evesrp.views.divisions.get_division_details (division_id=None, division=None)`

Generate a page showing the details of a division.

Shows which groups and individuals have been granted permissions to each division.

Only accessible to administrators.

Parameters `division_id (int)` – The ID number of the division

`evesrp.views.divisions.list_transformers (division_id, attribute=None)`

API method to get a list of transformers for a division.

Parameters

- **int** (`division_id`) – the ID of the division to look up
- **str** (`attribute`) – a specific attribute to look up. Optional.

Returns JSON

`evesrp.views.divisions.modify_division (division_id)`

Dispatches modification requests to the specialized view function for that operation.

`evesrp.views.divisions.permissions()`

Show a page listing all divisions.

`evesrp.views.divisions.transformer_choices (attr)`

List of tuples enumerating attributes that can be transformed/linked. Mainly used as the choices argument to `SelectField`

Requests

class `evesrp.views.requests.PayoutListing`

A special view made for quickly processing payouts for requests.

class `evesrp.views.requests.PermissionRequestListing (permissions, statuses, title=None)`

Show all requests that the current user has permissions to access.

This is used for the various permission-specific views.

__init__ (`permissions, statuses, title=None`)

Create a `PermissionRequestListing` for the given permissions and statuses.

Parameters

- **permissions** (*tuple*) – The permissions to filter by
- **statuses** (*tuple*) – A tuple of valid statuses for requests to be in

class `evesrp.views.requests.PersonalRequests`

Shows a list of all personally submitted requests and divisions the user has permissions in.

It will show all requests the current user has submitted.

class `evesrp.views.requests.RequestListing`

Abstract class for lists of `Requests`.

Subclasses will be able to respond to both normal HTML requests as well as to API requests with JSON.

decorators = [<function login_required at 0x7f796028b7b8>, <function varies.<locals>.vary_decorator at 0x7f795f3d5f3d>]

Decorators to apply to the view functions

dispatch_request (*filters=''*, ***kwargs*)

Returns the response to requests.

Part of the `flask.views.View` interface.

requests (*filters*)

Returns a list `Requests` belonging to the specified `Division`, or all divisions if `None`.

Returns `Requests`

Return type iterable

template = 'requests_list.html'

The template to use for listing requests

class `evesrp.views.requests.ValidKillmail` (*mail_class*, ***kwargs*)

Custom `:py:class:`~.Field`` validator that checks if any `Killmail` accepts the given URL.

`evesrp.views.requests.get_killmail_validators()`

Get a list of `ValidKillmails` for each killmail source.

This method is used to delay accessing `current_app` until we're in a request context. :returns: a list of `ValidKillmails` :rtype list:

`evesrp.views.requests.get_request_details` (*request_id=None*, *srp_request=None*)

Handles responding to all of the `Request` detail functions.

The various modifier functions all depend on this function to create the actual response content. Only one of the arguments is required. The `srp_request` argument is a convenience to other functions calling this function that have already retrieved the request.

Parameters

- **request_id** (*int*) – the ID of the request.
- **srp_request** (`Request`) – the request.

`evesrp.views.requests.modify_request` (*request_id*)

Handles POST requests that modify `Requests`.

Because of the numerous possible forms, this function bounces execution to a more specific function based on the form's "id_" field.

Parameters **request_id** (*int*) – the ID of the request.

`evesrp.views.requests.register_perm_request_listing` (*app*, *endpoint*, *path*, *permissions*, *statuses*, *title=None*)

Utility function for creating `PermissionRequestListing` views.

Parameters

- **app** (`flask.Flask`) – The application to add the view to
- **endpoint** (*str*) – The name of the view
- **path** (*str*) – The URL path for the view
- **permissions** (*tuple*) – Passed to `PermissionRequestListing.__init__()`
- **statuses** (*iterable*) – Passed to `PermissionRequestListing.__init__()`

`evesrp.views.requests.submit_request` ()

Submit a `Request`.

Displays a form for submitting a request and then processes the submitted information. Verifies that the user has the appropriate permissions to submit a request for the chosen division and that the killmail URL given is valid. Also enforces that the user submitting this requests controls the character from the killmail and prevents duplicate requests.

`evesrp.views.requests.url_for_page(pager, page_num)`

Utility method used in Jinja templates.

Models

class `evesrp.models.ActionType`

An Enum for representing the types of `Actions` performed on a `Request` in addition to the `status` of a `Request`.

statuses

A `frozenset` of all of the single `ActionType` members that also double as statuses for `Requests`.

finalized

A `frozenset` of the `ActionTypes` that are terminal states for a `Request` (`paid` and `rejected`).

pending

A `frozenset` of `ActionTypes` for `Requests` that require further action to be put in a `finalized` state.

approved = <approved>

Status for a request that has been evaluated and is awaiting payment.

comment = <comment>

A special type of `Action` representing a comment made on the request.

evaluating = <evaluating>

Status for a request being evaluated.

incomplete = <incomplete>

Status for a request that is missing details and needs further action.

paid = <paid>

Status for a request that has been paid. This is a terminating state.

rejected = <rejected>

Status for a requests that has been rejected. This is a terminating state.

exception `evesrp.models.ActionError`

Error raised for invalid state changes for a `Request`.

class `evesrp.models.Action(request, user, note=None, type_=None)`

Bases: `flask_sqlalchemy.Model`, `evesrp.util.models.AutoID`,
`evesrp.util.models.Timestamped`, `evesrp.util.models.AutoName`

Actions change the state of a `Request`.

`Requests` enforce permissions when actions are added to them. If the user adding the action does not have the appropriate `Permissions` in the request's `Division`, an `ActionError` will be raised.

With the exception of the `comment` action (which just adds text to a request), actions change the `status` of a `Request`.

note

Any additional notes for this action.

request

The [Request](#) this action applies to.

type_

The action be taken. See [ActionType](#) for possible values.

user

The [User](#) who made this action.

class `evesrp.models.ModifierError`

Error raised when a modification is attempted to a [Request](#) when it's in an invalid state.

class `evesrp.models.Modifier(request, user, note, value)`

Bases: `flask_sqlalchemy.Model`, `evesrp.util.models.AutoID`,
`evesrp.util.models.Timestamped`, `evesrp.util.models.AutoName`

Modifiers apply bonuses or penalties to Requests.

This is an abstract base class for the pair of concrete implementations. Modifiers can be voided at a later date. The user who voided a modifier and when it was voided are recorded.

[Requests](#) enforce permissions when modifiers are added. If the user adding a modifier does not have the appropriate [Permissions](#) in the request's [Division](#), a [ModifierError](#) will be raised.

voided

Boolean of whether this modifier has been voided or not.

This property is available as a [hybrid_property](#), so it can be used natively in SQLAlchemy queries.

note

Any notes explaining this modification.

request

The [Request](#) this modifier applies to.

user

The [User](#) who added this modifier.

void(user)

Mark this modifier as void.

Parameters `user` ([User](#)) – The user voiding this modifier

voided_timestamp

If this modifier has been voided, this will be the timestamp of when it was voided.

voided_user

The [User](#) who voided this modifier if it has been voided.

class `evesrp.models.AbsoluteModifier(request, user, note, value)`

Subclass of [Modifier](#) for representing absolute modifications.

Absolute modifications are those that are not dependent on the value of `Request.base_payout`.

value

How much ISK to add or remove from the payout

class `evesrp.models.RelativeModifier(request, user, note, value)`

Subclass of [Modifier](#) for representing relative modifiers.

Relative modifiers depend on the value of `Modifier.base_payout` to calculate their effect.

value

What percentage of the payout to add or remove

```
class evesrp.models.Request (submitter, details, division, killmail, **kwargs)
    Bases: flask_sqlalchemy.Model, evesrp.util.models.AutoID,
           evesrp.util.models.Timestamped, evesrp.util.models.AutoName
```

Requests represent SRP requests.

payout

The total payout of this request taking all active `modifiers` into account.

In calculating the total payout, all `absolute modifiers` along with the `base_payout` are summed. This is then multiplied by the sum of all of the `relative modifiers` plus 1. This property is a read-only `hybrid_property`, so it can be used natively in SQLAlchemy queries.

finalized

Boolean of if this request is in a `finalized` state. Also a read-only `hybrid_property` so it can be used natively in SQLAlchemy queries.

```
__init__ (submitter, details, division, killmail, **kwargs)
    Create a Request.
```

Parameters

- **submitter** (`User`) – The user submitting this request
- **details** (`str`) – Supporting details for this request
- **division** (`Division`) – The division this request is being submitted to
- **killmail** (`Killmail`) – The killmail this request pertains to

actions

A list of `Actions` that have been applied to this request, sorted in the order they were applied.

alliance

The alliance of the `pilot` at the time of the killmail.

base_payout

The base payout for this request.

This value is clamped to a lower limit of 0. It can only be changed when this request is in an `evaluating` state, or else a `ModifierError` will be raised.

constellation

The constellation this loss occurred in.

corporation

The corporation of the `pilot` at the time of the killmail.

details

Supporting information for the request.

division

The `Division` this request was submitted to.

kill_timestamp

The date and time of when the ship was destroyed.

killmail_url

The URL of the source killmail.

modifiers

A list of all `Modifiers` that have been applied to this request, regardless of whether they have been voided or not. They're sorted in the order they were added.

payout

The payout for this requests taking into account all active modifiers.

pilot

The `Pilot` who was the victim in the killmail.

region

The region this loss occured in.

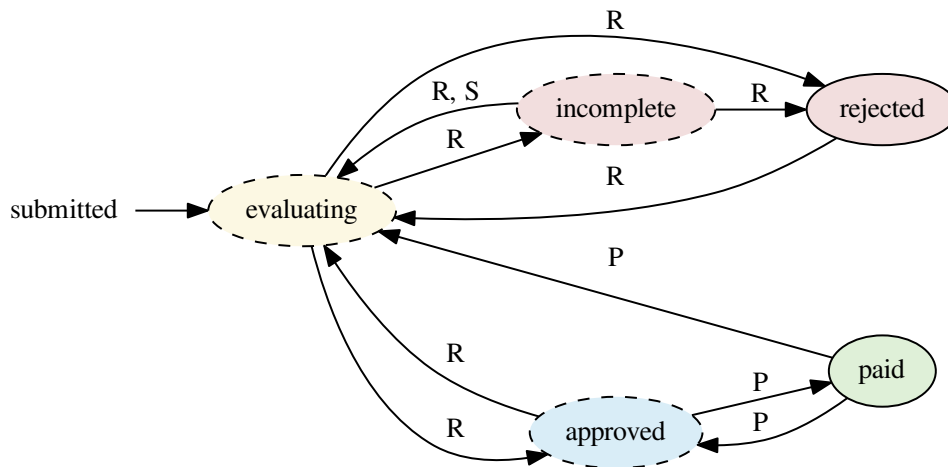
ship_type

The type of ship that was destroyed.

status

This attribute is automatically kept in sync as `Actions` are added to the request. It should not be set otherwise.

At the time an `Action` is added to this request, the type of action is checked and the state diagram below is enforced. If the action is invalid, an `ActionError` is raised.



R means a reviewer can make that change, S means the submitter can make that change, and P means payers can make that change. Solid borders are terminal states.

submitter

The `User` who submitted this request.

system

The solar system this loss occured in.

transformed

Get a special HTML representation of an attribute.

Divisions can have a transformer defined on various attributes that output a URL associated with that attribute. This property provides easy access to the output of any transformed attributes on this request.

valid_actions (*user*)

Get valid actions (besides comment) the given user can perform.

Javascript

The following documentation is directed towards people developing the front-end for EVE-SRP. These functions should not be used by end-users, and are purely an implementation detail.

Utilities

month (*month_int*)

Convert an integer representing a month to the three letter abbreviation.

Arguments

- **month_int** (*int*) – An integer (0-11) representing a month.

Returns The three letter abbreviation for that month.

Return type string

padNum (*num*, *width*)

Pad a number with leading 0s to the given width.

Arguments

- **num** (*int*) – The number to pad.
- **width** (*int*) – The width to pad *num* to.

Returns *num* padded to *width* with 0s.

Return type string

pageNumbers (*num_pages*, *current_page* [, *options*])

Return an array of page numbers, skipping some of them as configured by the options argument. This function should be functionally identical to Flask-SQLAlchemy's `Pagination.iter_pages` (including in default arguments). One deviation is that this function uses 0-indexed page numbers instead of 1-indexed, to ease compatibility with PourOver. Skipped numbers are represented by `null`.

Arguments

- **num_pages** (*int*) – The total number of pages.
- **current_page** (*int*) – The index of the current page.
- **options** – An object with vonfiguration values for where to skip numbers. Keys are `left_edge`, `left_current`, `right_current`, and `right_edge`. The default values are 2, 2, 5 and 2 respectively.

Returns The page numbers to be show, in order.

Return type An array on integers (and `null`).

pager_a_click (*ev*)

Event callback for pager links. It intercepts the event and changes the current PourOver view to reflect the new page.

Arguments

- **ev** (*event*) – The event object.

PourOver

class RequestsView (*name, collection*)

An extension of `PourOver.View` with a custom `render` function recreating a table of `Requests` with the associated pager.

addRequestSorts (*collection*)

Add sorts for `Request` attributes to the given `PourOver.Collection`.

Arguments

- **collection** (*PourOver.Collection*) – A collection of requests.

addRequestFilters (*collection*)

Add filters for `Request` attributes to the given `PourOver.Collection`.

Arguments

- **collection** (*PourOver.Collection*) – A collection of requests.

Indices and tables

- *genindex*
- *modindex*
- *search*

e

- `evesrp.auth`, [26](#)
- `evesrp.auth.bravecore`, [23](#)
- `evesrp.auth.everso`, [25](#)
- `evesrp.auth.j4oauth`, [25](#)
- `evesrp.auth.models`, [27](#)
- `evesrp.auth.oauth`, [23](#)
- `evesrp.auth.testauth`, [23](#)
- `evesrp.auth.testoauth`, [26](#)
- `evesrp.killmail`, [33](#)
- `evesrp.models`, [38](#)
- `evesrp.views`, [35](#)
- `evesrp.views.divisions`, [36](#)
- `evesrp.views.login`, [35](#)
- `evesrp.views.requests`, [36](#)

Symbols

__init__() (evesrp.auth.AuthMethod method), 27
 __init__() (evesrp.auth.bravecore.BraveCore method), 23
 __init__() (evesrp.auth.j4oauth.J4OAuth method), 25
 __init__() (evesrp.auth.models.Permission method), 29
 __init__() (evesrp.auth.models.Pilot method), 29
 __init__() (evesrp.auth.oauth.OAuthMethod method), 24
 __init__() (evesrp.auth.testauth.TestAuth method), 23
 __init__() (evesrp.auth.testoauth.TestOAuth method), 26
 __init__() (evesrp.killmail.ESIMail method), 34
 __init__() (evesrp.killmail.Killmail method), 34
 __init__() (evesrp.killmail.RequestsSessionMixin method), 35
 __init__() (evesrp.models.Request method), 40
 __init__() (evesrp.views.requests.PermissionRequestListing method), 36
 __iter__() (evesrp.killmail.Killmail method), 34

A

AbsoluteModifier (class in evesrp.models), 39
 Action (class in evesrp.models), 38
 ActionError, 38
 actions (evesrp.auth.models.User attribute), 28
 actions (evesrp.models.Request attribute), 40
 ActionType (class in evesrp.models), 38
 add_division() (in module evesrp.views.divisions), 36
 addRequestFilters() (built-in function), 43
 addRequestSorts() (built-in function), 43
 admin (evesrp.auth.models.User attribute), 28
 admin (evesrp.auth.PermissionType attribute), 26
 all (evesrp.auth.PermissionType attribute), 26
 alliance (evesrp.killmail.Killmail attribute), 33
 alliance (evesrp.models.Request attribute), 40
 alliance_id (evesrp.killmail.Killmail attribute), 33
 APIKey (class in evesrp.auth.models), 29
 approved (evesrp.models.ActionType attribute), 38
 attribute_name (evesrp.auth.models.TransformerRef attribute), 30
 audit (evesrp.auth.PermissionType attribute), 26
 auth_method_login() (in module evesrp.views.login), 35

AuthMethod (class in evesrp.auth), 27
 authmethod (evesrp.auth.models.Entity attribute), 28

B

base_payout (evesrp.models.Request attribute), 40
 BraveCore (class in evesrp.auth.bravecore), 23

C

comment (evesrp.models.ActionType attribute), 38
 constellation (evesrp.killmail.Killmail attribute), 34
 constellation (evesrp.killmail.LocationMixin attribute), 35
 constellation (evesrp.models.Request attribute), 40
 content (evesrp.auth.models.Note attribute), 29
 corp (evesrp.killmail.Killmail attribute), 33
 corp_id (evesrp.killmail.Killmail attribute), 33
 corporation (evesrp.models.Request attribute), 40

D

decorators (evesrp.views.requests.RequestListing attribute), 36
 description (evesrp.killmail.Killmail attribute), 34
 details (evesrp.models.Request attribute), 40
 dispatch_request() (evesrp.views.requests.RequestListing method), 37
 Division (class in evesrp.auth.models), 30
 division (evesrp.auth.models.Permission attribute), 29
 division (evesrp.auth.models.TransformerRef attribute), 30
 division (evesrp.models.Request attribute), 40
 division_permissions (evesrp.auth.models.Division attribute), 30
 domain (evesrp.killmail.ZKillmail attribute), 34

E

elevated (evesrp.auth.PermissionType attribute), 26
 Entity (class in evesrp.auth.models), 27
 entity (evesrp.auth.models.Permission attribute), 30
 entity_permissions (evesrp.auth.models.Entity attribute), 28

ESIMail (class in `evesrp.killmail`), 34
evaluating (`evesrp.models.ActionType` attribute), 38
`evesrp.auth` (module), 26
`evesrp.auth.bravecore` (module), 23
`evesrp.auth.everso` (module), 25
`evesrp.auth.j4oauth` (module), 25
`evesrp.auth.models` (module), 27
`evesrp.auth.oauth` (module), 23
`evesrp.auth.testauth` (module), 23
`evesrp.auth.testoauth` (module), 26
`evesrp.killmail` (module), 33
`evesrp.models` (module), 38
`evesrp.views` (module), 35
`evesrp.views.divisions` (module), 36
`evesrp.views.login` (module), 35
`evesrp.views.requests` (module), 36
EveSSO (class in `evesrp.auth.everso`), 25

F

finalized (`evesrp.models.ActionType` attribute), 38
finalized (`evesrp.models.Request` attribute), 40
form() (`evesrp.auth.AuthMethod` method), 27

G

get_division_details() (in module `evesrp.views.divisions`), 36
get_groups() (`evesrp.auth.everso.EveSSO` method), 25
get_groups() (`evesrp.auth.oauth.OAuthMethod` method), 24
get_id() (`evesrp.auth.models.User` method), 28
get_killmail_validators() (in module `evesrp.views.requests`), 37
get_pilots() (`evesrp.auth.oauth.OAuthMethod` method), 24
get_request_details() (in module `evesrp.views.requests`), 37
get_user() (`evesrp.auth.oauth.OAuthMethod` method), 24
Group (class in `evesrp.auth.models`), 29
groups (`evesrp.auth.models.User` attribute), 28

H

has_permission() (`evesrp.auth.models.Entity` method), 28
hex_key (`evesrp.auth.models.APIKey` attribute), 29

I

incomplete (`evesrp.models.ActionType` attribute), 38
index() (in module `evesrp.views`), 35
is_active (`evesrp.auth.models.User` attribute), 28
is_admin() (`evesrp.auth.oauth.OAuthMethod` method), 25
is_anonymous (`evesrp.auth.models.User` attribute), 28
is_authenticated (`evesrp.auth.models.User` attribute), 28

J

J4OAuth (class in `evesrp.auth.j4oauth`), 25

K

key (`evesrp.auth.models.APIKey` attribute), 29
kill_id (`evesrp.killmail.Killmail` attribute), 33
kill_timestamp (`evesrp.models.Request` attribute), 40
Killmail (class in `evesrp.killmail`), 33
killmail_url (`evesrp.models.Request` attribute), 40

L

list_transformers() (in module `evesrp.views.divisions`), 36
LocationMixin (class in `evesrp.killmail`), 35
login() (`evesrp.auth.AuthMethod` method), 27
login() (in module `evesrp.views.login`), 35
login_loader() (in module `evesrp.views.login`), 35
login_user() (`evesrp.auth.AuthMethod` static method), 27
logout() (in module `evesrp.views.login`), 35

M

Modifier (class in `evesrp.models`), 39
ModifierError (class in `evesrp.models`), 39
modifiers (`evesrp.models.Request` attribute), 40
modify_division() (in module `evesrp.views.divisions`), 36
modify_request() (in module `evesrp.views.requests`), 37
month() (built-in function), 42

N

name (`evesrp.auth.models.Division` attribute), 30
name (`evesrp.auth.models.Entity` attribute), 28
name (`evesrp.auth.models.Pilot` attribute), 29
Note (class in `evesrp.auth.models`), 29
note (`evesrp.models.Action` attribute), 38
note (`evesrp.models.Modifier` attribute), 39
noter (`evesrp.auth.models.Note` attribute), 29

O

OAuthMethod (class in `evesrp.auth.oauth`), 24

P

padNum() (built-in function), 42
pageNumbers() (built-in function), 42
pager_a_click() (built-in function), 42
paid (`evesrp.models.ActionType` attribute), 38
pay (`evesrp.auth.PermissionType` attribute), 26
payout (`evesrp.models.Request` attribute), 40
PayoutListing (class in `evesrp.views.requests`), 36
pending (`evesrp.models.ActionType` attribute), 38
Permission (class in `evesrp.auth.models`), 29
permission (`evesrp.auth.models.Permission` attribute), 30
PermissionRequestListing (class in `evesrp.views.requests`), 36
permissions (`evesrp.auth.models.Division` attribute), 30
permissions (`evesrp.auth.models.Group` attribute), 29
permissions() (in module `evesrp.views.divisions`), 36
PermissionType (class in `evesrp.auth`), 26

PersonalRequests (class in `evesrp.views.requests`), 36
 Pilot (class in `evesrp.auth.models`), 28
 pilot (`evesrp.killmail.Killmail` attribute), 33
 pilot (`evesrp.models.Request` attribute), 41
 pilot_id (`evesrp.killmail.Killmail` attribute), 33
 pilots (`evesrp.auth.models.User` attribute), 28

R

refresh() (`evesrp.auth.AuthMethod` method), 27
 refresh() (`evesrp.auth.oauth.OAuthMethod` method), 25
 region (`evesrp.killmail.Killmail` attribute), 34
 region (`evesrp.killmail.LocationMixin` attribute), 35
 region (`evesrp.models.Request` attribute), 41
 register_perm_request_listing() (in module `evesrp.views.requests`), 37
 rejected (`evesrp.models.ActionType` attribute), 38
 RelativeModifier (class in `evesrp.models`), 39
 Request (class in `evesrp.models`), 39
 request (`evesrp.models.Action` attribute), 38
 request (`evesrp.models.Modifier` attribute), 39
 RequestListing (class in `evesrp.views.requests`), 36
 requests (`evesrp.auth.models.Division` attribute), 30
 requests (`evesrp.auth.models.Pilot` attribute), 29
 requests (`evesrp.auth.models.User` attribute), 28
 requests() (`evesrp.views.requests.RequestListing` method), 37
 requests_session (`evesrp.killmail.RequestsSessionMixin` attribute), 34
 RequestsSessionMixin (class in `evesrp.killmail`), 34
 RequestsView() (class), 43
 review (`evesrp.auth.PermissionType` attribute), 26

S

safe_name (`evesrp.auth.AuthMethod` attribute), 27
 ship (`evesrp.killmail.Killmail` attribute), 33
 ship (`evesrp.killmail.ShipNameMixin` attribute), 35
 ship_id (`evesrp.killmail.Killmail` attribute), 33
 ship_type (`evesrp.models.Request` attribute), 41
 ShipNameMixin (class in `evesrp.killmail`), 35
 status (`evesrp.models.Request` attribute), 41
 statuses (`evesrp.models.ActionType` attribute), 38
 submit (`evesrp.auth.PermissionType` attribute), 27
 submit_divisions() (`evesrp.auth.models.User` method), 28
 submit_request() (in module `evesrp.views.requests`), 37
 submitter (`evesrp.models.Request` attribute), 41
 system (`evesrp.killmail.Killmail` attribute), 34
 system (`evesrp.killmail.LocationMixin` attribute), 35
 system (`evesrp.models.Request` attribute), 41
 system_id (`evesrp.killmail.Killmail` attribute), 34

T

template (`evesrp.views.requests.RequestListing` attribute), 37
 TestAuth (class in `evesrp.auth.testauth`), 23

TestOAuth (class in `evesrp.auth.testoauth`), 26
 timestamp (`evesrp.killmail.Killmail` attribute), 33
 transformed (`evesrp.models.Request` attribute), 41
 transformer (`evesrp.auth.models.TransformerRef` attribute), 30
 transformer_choices() (in module `evesrp.views.divisions`), 36
 TransformerRef (class in `evesrp.auth.models`), 30
 transformers (`evesrp.auth.models.Division` attribute), 30
 type_ (`evesrp.models.Action` attribute), 39

U

url (`evesrp.killmail.Killmail` attribute), 33
 url_for_page() (in module `evesrp.views.requests`), 38
 User (class in `evesrp.auth.models`), 28
 user (`evesrp.auth.models.APIKey` attribute), 29
 user (`evesrp.auth.models.Note` attribute), 29
 user (`evesrp.auth.models.Pilot` attribute), 29
 user (`evesrp.models.Action` attribute), 39
 user (`evesrp.models.Modifier` attribute), 39
 users (`evesrp.auth.models.Group` attribute), 29

V

valid_actions() (`evesrp.models.Request` method), 41
 ValidKillmail (class in `evesrp.views.requests`), 37
 value (`evesrp.killmail.Killmail` attribute), 33
 value (`evesrp.models.AbsoluteModifier` attribute), 39
 value (`evesrp.models.RelativeModifier` attribute), 39
 verified (`evesrp.killmail.Killmail` attribute), 34
 view() (`evesrp.auth.AuthMethod` method), 27
 view() (`evesrp.auth.oauth.OAuthMethod` method), 25
 void() (`evesrp.models.Modifier` method), 39
 voided (`evesrp.models.Modifier` attribute), 39
 voided_timestamp (`evesrp.models.Modifier` attribute), 39
 voided_user (`evesrp.models.Modifier` attribute), 39

Z

ZKillmail (class in `evesrp.killmail`), 34