django-esi

Alliance Auth devs

Apr 10, 2024

CONTENTS:

1 Operations Guide **3**
- 1.1 Installation . . . 3
- 1.2 Upgrade . . . 4
- 1.3 Settings . . . 4

2 Developer Guide **7**
- 2.1 Usage in views . . . 7
- 2.2 Accessing ESI . . . 8
- 2.3 User Agent header . . . 11
- 2.4 Cleaning the database . . . 12
- 2.5 Advanced Features . . . 13
- 2.6 Exploring ESI Endpoints . . . 14

3 API **17**
- 3.1 clients . . . 17
- 3.2 decorators . . . 18
- 3.3 errors . . . 19
- 3.4 models . . . 19
- 3.5 managers . . . 21

4 Indices and tables **23**

Python Module Index **25**

Index **27**

django-esi is a Django app that provides easy access to the EVE Swagger Interface (ESI) for Django sites.

CHAPTER

ONE

OPERATIONS GUIDE

The operations guide describes how to install, configure and maintain *django-esi*.

1.1 Installation

To install django-esi into your Django project please follow the these steps:

1.1.1 Step 1: Install the latest version directly from PyPI

```
pip install django-esi
```

1.1.2 Step 2: Add `esi` to your `INSTALLED_APPS` setting

```
INSTALLED_APPS += [
    # other apps
    'esi',
    # other apps
]
```

1.1.3 Step 3: Include the esi urlconf in your project's urls

```
url(r'^sso/', include('esi.urls', namespace='esi')),
```

1.1.4 Step 4: Register an application with the EVE Developers site

If your application requires scopes, select **Authenticated API Access** and register all possible scopes your app can request. Otherwise **Authentication Only** will suffice.

Set the **Callback URL** to `https://example.com/sso/callback`

1.1.5 Step 4: Add SSO client settings to your project settings

```
ESI_SSO_CLIENT_ID = "my client id"
ESI_SSO_CLIENT_SECRET = "my client secret"
ESI_SSO_CALLBACK_URL = "https://example.com/sso/callback"
```

1.1.6 Step 5: Run migrations to create models

```
python manage.py migrate
```

1.2 Upgrade

To update an existing installation please first make sure that you are in your virtual environment and in the main project folder (the one that has `manage.py`). Then run the following commands one by one:

```
pip install -U django-esi
```

```
python manage.py migrate
```

```
python manage.py collectstatic
```

Finally restart your Django application, e.g. by restarting your supervisors.

1.3 Settings

Django-esi can be configured through settings by adding them to your Django settings file. Here is the list of the most commonly used settings:

1.3.1 Required settings

Required settings need to be set in order for django-esi to function.

ESI_SSO_CALLBACK_URL = 'http://localhost:8000'

Callback for your Eve Online SSO app. REQUIRED.

ESI_SSO_CLIENT_ID = 'test-dummy'

Client ID of your Eve Online SSO app. REQUIRED.

ESI_SSO_CLIENT_SECRET = 'test-dummy'

Client Secret of your Eve Online SSO app. REQUIRED.

Hint: These settings can be left blank if DEBUG is set to True.

1.3.2 Optional settings

Optional settings will use the documented default if they are not set.

ESI_ALWAYS_CREATE_TOKEN = False

Enable to force new token creation every callback.

ESI_API_DATASOURCE = 'tranquility'

Change these to switch to Singularity.

ESI_API_VERSION = 'latest'

Change this to access different revisions of the ESI API by default

ESI_CACHE_RESPONSE = True

Disable to stop caching endpoint responses.

ESI_CONNECTION_ERROR_MAX_RETRIES = 3

Max retries on failed connections.

ESI_CONNECTION_POOL_MAXSIZE = 10

Max size of the connection pool.

Increase this setting if you hav more parallel threads connected to ESI at the same time.

ESI_DEBUG_RESPONSE_CONTENT_LOGGING = True

Enable/Disable logging of ESI response contents.

ESI_INFO_LOGGING_ENABLED = False

Enable/disable verbose info logging.

ESI_LOG_LEVEL_LIBRARIES = 'INFO'

Set log level for libraries like bravado and urlib3.

ESI_REQUESTS_CONNECT_TIMEOUT = 5

Default connection timeouts for all requests to ESI.

Can temporarily overwritten with by passing `timeout` with `result()`

ESI_REQUESTS_READ_TIMEOUT = 30

Default read timeouts for all requests to ESI.

Can temporarily overwritten with by passing `timeout` with `result()`

ESI_SERVER_ERROR_BACKOFF_FACTOR = 0.2

Backoff factor for retries on server error.

ESI_SERVER_ERROR_MAX_RETRIES = 3

Max retries on server errors.

ESI_USER_CONTACT_EMAIL = None

Contact email address of server owner.

This will be included in the User-Agent header of every request.

See also:

For a list of all settings please see `esi.app_settings`.

CHAPTER

TWO

DEVELOPER GUIDE

The developer guide describes how to develop apps with *django-esi*.

2.1 Usage in views

2.1.1 Single token

When views require a token, wrap with the `token_required` decorator and accept a `token` arg:

```
from esi.decorators import token_required

@token_required(scopes="esi-characters.read_medals.v1")
def my_view(request, token):
    # my code
```

This will prompt the user to either select a token from their current ones, or if none exist create a new one via SSO.

To specify scopes, add either a list of names or a space-delimited string:

```
@token_required(scopes=['esi-location.read_ship_type.v1', 'esi-location.read_location.v1
↪'])
@token_required(scopes='esi-location.read_ship_type.v1 esi-location.read_location.v1')
```

2.1.2 New token

To require a new token, such as for logging in, add the `new` argument:

```
@token_required(new=True)
```

2.1.3 Multiple tokens

To request all of a user's tokens which have the required scopes, wrap instead with the `tokens_required` decorator and accept a `tokens` arg:

```
@tokens_required(scopes='esi-location.read_ship_type.v1')
def my_view(request, tokens):
    # my code
```

This skips prompting for token selection and instead passes that responsibility to the view. Tokens are provided as a queryset.

2.1.4 Single use token

It is also possible to request a token for single use. Single use tokens do not require a user to be logged in and are only available to the current view.

```
from esi.decorators import single_use_token

@single_use_token(scopes=['publicData'])
my_view(request, token):
    # my code
```

See also:

See API section *decorators* for more details on all provided decorators.

2.2 Accessing ESI

django-esi provides a convenience wrapper around the bravado SwaggerClient.

2.2.1 New approach for getting a client object

All access to ESI happens through a client object that is automatically generated for you and contains all of ESI's routes. The new and **recommended** way of getting that client object is through a single provider instance from the *EsiClientProvider* class.

The new provider approach has two main advantages: First, creating a new client is slow (e.g. can takes up to 5 seconds). So, for maximum performance you want to avoid creating multiple clients in your app. Using the provider automatically ensures this.

Second, the previous approach of creating multiple clients can cause memory leaks. Especially when used in concurrent environment (e.g. threads or celery tasks), where each worker is creating it's own client.

2.2.2 Example for creating a provider

The provider needs to be instantiated at "import time", so it must be defined in the global scope of a module.

```
from esi.clients import EsiClientProvider

# create your own provider
esi = EsiClientProvider()

def main():
    # do stuff with your provider
```

If you need to use the provider in several module than a good pattern is to define it in it's own module, e.g. `providers.py`, and then import the provider instance into all other modules that need an ESI client.

2.2.3 Using public endpoints

Here is a complete example how to use a public endpoint. Public endpoints can in general be accessed without any authentication.

```
from esi.clients import EsiClientProvider

# create your own provider
esi = EsiClientProvider()

def main():
    # call the endpoint
    result = esi.client.Status.get_status().results()

    # ... do stuff with the data
    print(result)
```

2.2.4 Using authenticated endpoints

Non-public endpoints will require authentication. You will therefore need to provide a valid access token with your request.

The following example shows how to retrieve data from a non-public endpoint using an already existing token in your database.

See also:

See also the section *Usage in views* on how to create tokens in your app.

```
from esi.clients import EsiClientProvider
from esi.models import Token

# create your own provider
esi = EsiClientProvider()

def main():
    character_id = 1234
    required_scopes = ['esi-characters.read_notifications.v1']

    # get a token
    token = Token.get_token(character_id, required_scopes)

    # call the endpoint
    notifications = esi.client.Character.get_characters_character_id_notifications(
        # required parameter for endpoint
        character_id = character_id,
        # provide a valid access token, which wil be refresh the token if required
        token = token.valid_access_token()
    ).results()

    # ... do stuff with the data
```

2.2.5 results() vs. result()

django-esi offers two similar methods for requesting the response from an endpoint: results() and result(). Here is a quick overview how they differ:

Topic	results()	result()
Paging	Automatically returns all pagesif they are more than one	Only returns the fist page or therequested page(when specified with page parameter)
Headers	Returns the headers forthe last retrieved page	Returns the headers for the first /requested page
Backwards compatibility	New feature in 2.0	Works mostly as in 1.6

In general we recommend to use results(), so you don't have to worry about paging. Nevertheless, result() gives you more direct control of your API request and has it's uses, e.g when you are only interested in the first page and do not want to wait for all pages to download from the API.

2.2.6 Getting localized responses from ESI

Some ESI endpoints support localization, which means they are able to return the content localized in one of the supported languages.

To retrieve localized content just provide the language code in your request. The following example will retrieve the type info for the Svipul in Korean:

```
result = (
    esi.client.Universe
    .get_universe_types_type_id(type_id=30004984, language='ko')
    .results()
)
```

A common use case it to retrieve localizations for all languages for the current request. For this django-esi provides the convenience method `results_localized()`. It substitutes `results()` and will return the response in all officially supported languages by default.

```
result = (
    esi.client.Universe
    .get_universe_types_type_id(type_id=30004984)
    .results_localized()
)
```

Alternatively you can pass the list of languages (as language code) that you are interested in:

```
result = (
    esi.client.Universe
    .get_universe_types_type_id(type_id=30004984)
    .results_localized(languages=['ko', 'de'])
)
```

2.2.7 Specifying resource versions

As explained on the EVE Developers Blog, it's best practice to call a specific version of the resource and allow the ESI router to map it to the correct route, being `legacy`, `latest` or `dev`.

Client initialization begins with a base swagger spec. By default this is the version defined in settings (`ESI_API_VERSION`), but can be overridden with an extra argument to the factory:

```
client = esi_client_factory(version='v4')
```

Only resources with the specified version number will be available. For instance, if you specify `v4` but `Universe` does not have a `v4` version, it will not be available to that specific client. Only `legacy`, `latest` and `dev` are guaranteed to have all resources available.

Individual resources are versioned and can be accessed by passing additional arguments to the factory:

```
client = esi_client_factory(Universe='v1', Character='v3')
```

A list of available resources is available on the EVE Swagger Interface browser. If the resource is not available with the specified version, an `AttributeError` will be raised.

This version of the resource replaces the resource originally initialized. If the requested base version does not have the specified resource, it will be added.

Note that only one old revision of each resource is kept available through the legacy route. Keep an eye on the deployment timeline for resource updates.

2.3 User Agent header

CCP asks developers to provide a "good User-Agent header" with all requests to ESI, so that CCP can identify which app the request belongs to and is able to contact the server owner running the app in case of any issues. This requirement is specified in the CCP's Developer Guidelines and detailed in the ESI guidelines.

Django-esi provides two features for setting the User-Agent header:

2.3.1 Application Info

You configure the User-Agent to represent your application by setting the `app_info_text` parameter when creating a client with *EsiClientProvider* or with *esi_client_factory*.

There is no official format for the User-Agent, but we would suggest including the distribution name (same as e.g. in your `setup.py`) and current version of your application like so:

```
"my-app v1.0.0"
```

Here is a complete example for defining an application string with your app:

```
from esi.clients import EsiClientProvider

esi = EsiClientProvider(app_info_text="my-app v1.0.0")
```

Hint: Spaces are used as delimiter in the User Agent, so your application name should not include any.

Note: If you do not define an application string, the application string used will be "`django-esi vX.Y.Z`".

2.3.2 Contact email

To enable CCP to contact the maintainer of a server that is using ESI it is important to specify a contact email. This can be done through the setting ESI_USER_CONTACT_EMAIL.

Example:

```
ESI_USER_CONTACT_EMAIL = "admin@example.com"
```

In case you are not hosting the app yourself, we would recommend including this setting in the installation guide for your app.

2.4 Cleaning the database

Two tasks are available:

- `cleanup_callbackredirect` removes all `CallbackRedirect` models older than a specified age (in seconds). Default is 300.
- `cleanup_token` checks all `Token` models, and if expired, attempts to refresh. If expired and cannot refresh, or fails to refresh, the model is deleted.

To schedule these automatically with celerybeat, add them to your settings.py `CELERYBEAT_SCHEDULE` dict like so:

```
from celery.schedules import crontab

CELERYBEAT_SCHEDULE = {
    ...
    'esi_cleanup_callbackredirect': {
        'task': 'esi.tasks.cleanup_callbackredirect',
        'schedule': crontab(hour='*/4'),
    },
    'esi_cleanup_token': {
        'task': 'esi.tasks.cleanup_token',
        'schedule': crontab(day_of_month='*/1'),
    },
}
```

Recommended intervals are four hours for callback redirect cleanup and daily for token cleanup (token cleanup can get quite slow with a large database, so adjust as needed). If your app does not require background token validation, it may be advantageous to not schedule the token cleanup task, instead relying on the validation check when using `@token_required` decorators or adding `.require_valid()` to the end of a query.

2.5 Advanced Features

2.5.1 Using a local spec file

Specifying resource versions introduces one major problem for shared code: not all resources nor all their operations are available on any given version. This can be addressed by shipping a copy of the versioned latest spec with your app. **This is the preferred method for deployment.**

To build a client using this local spec, pass an additional parameter `spec_file` which contains the path to your local swagger.json:

```
from esi.clients import EsiClientProvider

esi = EsiClientProvider(spec_file='/path/to/swagger.json')
```

For example, a swagger.json in the current file's directory would look like:

```
import os
from esi.clients import EsiClientProvider

SWAGGER_SPEC = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'swagger.json')
esi = EsiClientProvider(spec_file=SWAGGER_SPEC)
```

If a `spec_file` is specified all other versioning is unavailable: ensure you ship a spec with resource versions your app can handle.

2.5.2 Getting Response Data

Sometimes you may want to also get the internal response object from an ESI response. For example to inspect the response header. For that simply set the `request_config.also_return_response` to `True` and then call the endpoint. This works in the same way for both `.result()` and `.results()`

```
from esi.clients import EsiClientProvider
from esi.models import Token

# create your own provider
esi = EsiClientProvider()

def main():
    character_id = 1234
    required_scopes = ['esi-characters.read_notifications.v1']

    # get a token
    token = Token.get_token(character_id, required_scopes)

    # call the endpoint but don't request data.
    operation = esi.client.Character.get_characters_character_id_notifications(
        character_id = character_id,
        token = token.valid_access_token()
    )
```

(continues on next page)

(continued from previous page)

```
    # set to get the response as well
    operation.request_config.also_return_response = True

    # get your data
    notifications, response = operation.results()

    # ... do stuff with the data
    print(response.headers['Expires'])
```

2.5.3 Accessing alternate data sources

ESI data source can also be specified during client creation:

```
from esi.clients import EsiClientProvider
from esi.models import Token

# create your own provider
esi = EsiClientProvider(datasource='tranquility')
```

Currently the only available data source is `tranquility`, which is also the default The previously available datasource `singularity` is no longer available.

2.6 Exploring ESI Endpoints

The builtin Django shell allows you to explore the EVE ESI endpoints via django-esi, making it an easy way to find what methods are available for consumption.

2.6.1 Prerequisites

Prior to using the Django shell to explore django-esi, you must first *install and configure* django-esi in the Django project.

2.6.2 Getting Started

Open up a command line and navigate to the Django folder containing manage.py.

```
$ python manage.py shell
>>> from esi.clients import esi_client_factory
>>> c = esi_client_factory()
>>> print(dir( c ))
```

The above commands in the Django shell should show something like this:

```
['Alliance',
'Assets',
'Bookmarks',
...
```

(continues on next page)

(continued from previous page)

```
'Wallet',
'Wars']
```

2.6.3 Further Uses

Once it's working, you can explore further endpoints just by adding to the c variable.

```
>>> print(dir(c.Universe))
'get_universe_ancestries',
'get_universe_asteroid_belts_asteroid_belt_id',
...
'post_universe_ids',
'post_universe_names']
```

CHAPTER

THREE

API

This chapter contains the developer reference documentation of the public API for *django-esi*.

3.1 clients

class CachingHttpFuture(*future: FutureAdapter*, *response_adapter: Callable[[Any], IncomingResponse]*, *operation: Operation | None = None*, *request_config: RequestConfig | None = None*)

Extended wrapper for a FutureAdapter that returns a HTTP response and also supports caching.

This class contains the response for an ESI request with an ESI client.

result(***kwargs*) → Any | Tuple[Any, IncomingResponse]

Executes the request and returns the response from ESI. Response will include the requested / first page only if there are more pages available.

Parameters

- **timeout** – (optional) timeout for ESI request in seconds, overwrites default
- **retries** – (optional) max number of retries, overwrites default
- **language** – (optional) retrieve result for specific language
- **ignore_cache** – (optional) set to `True` to ignore response caching

Returns

Response from endpoint or a tuple with response from endpoint and an incoming response object containing additional meta data including the HTTP response headers

results(***kwargs*) → Any | Tuple[Any, IncomingResponse]

Executes the request and returns the response from ESI for the current route. Response will include all pages if there are more available.

Accepts same parameters in `kwargs` as *result()*

Returns

same as *result()*, but for multiple pages

results_localized(*languages: list = None*, ***kwargs*) → dict

Executes the request and returns the response from ESI for all default languages and pages (if any).

Accepts same parameters in `kwargs` as *result()* plus `languages`

Parameters

languages – (optional) list of languages to return instead of default languages

Returns
Dict of all responses with the language code as keys.

`class EsiClientProvider`(*datasource=None*, *spec_file=None*, *version=None*, *app_info_text=None*, ***kwargs*)

Class for providing a single ESI client instance for the whole app

Parameters

- **datasource** – Name of the ESI datasource to access.
- **spec_file** – Absolute path to a swagger spec file to load.
- **version** – Base ESI API version. Accepted values are 'legacy', 'latest',
- **app_info_text** – Text identifying the application using ESI which will be included in the User-Agent header. Should contain name and version of the application using ESI. e.g. *"my-app v1.0.0"*. Note that spaces are used as delimiter.
- **kwargs** – Explicit resource versions to build, in the form Character='v4'. Same values accepted as version.

If a spec_file is specified, specific versioning is not available. Meaning the version and resource version kwargs are ignored in favour of the versions available in the spec_file.

`esi_client_factory`(*token=None*, *datasource: str = None*, *spec_file: str = None*, *version: str = None*, *app_info_text: str = None*, ***kwargs*) → SwaggerClient

Generate a new ESI client.

Parameters

- **token** (esi.models.Token) – used to access authenticated endpoints.
- **datasource** – Name of the ESI datasource to access.
- **spec_file** – Absolute path to a swagger spec file to load.
- **version** – Base ESI API version. Accepted values are 'legacy', 'latest',
- **app_info_text** – Text identifying the application using ESI which will be included in the User-Agent header. Should contain name and version of the application using ESI. e.g. *"my-app v1.0.0"*. Note that spaces are used as delimiter.
- **kwargs** – Explicit resource versions to build, in the form Character='v4'. Same values accepted as version.

If a spec_file is specified, specific versioning is not available. Meaning the version and resource version kwargs are ignored in favour of the versions available in the spec_file.

Returns
New ESI client

3.2 decorators

`single_use_token`(*scopes=''*, *new=False*)

Decorator for views which supplies a single use token granted via sso login regardless of login state. Same parameters as tokens_required.

`token_required`(*scopes=''*, *new=False*)

Decorator for views which supplies a single, user-selected token for the view to process. Same parameters as tokens_required.

tokens_required(*scopes=''*, *new=False*)

Decorator for views to request an ESI Token. Accepts required scopes as a space-delimited string or list of strings of scope names. Can require a new token to be retrieved by SSO. Returns a QueryDict of Tokens.

3.3 errors

exception DjangoEsiException

exception IncompleteResponseError

exception NotRefreshableTokenError

exception TokenError

exception TokenExpiredError

exception TokenInvalidError

3.4 models

class Token(**args*, ***kwargs*)

EVE Swagger Interface Access Token

Contains information about the authenticating character and scopes granted to this token. Contains the access token required for ESI authentication as well as refreshing.

Parameters

- **id** (*AutoField*) – Primary key: ID
- **created** (*DateTimeField*) – Created
- **access_token** (*TextField*) – Access token. The access token granted by SSO.
- **refresh_token** (*TextField*) – Refresh token. A re-usable token to generate new access tokens upon expiry.
- **character_id** (*IntegerField*) – Character id. The ID of the EVE character who authenticated by SSO.
- **character_name** (*CharField*) – Character name. The name of the EVE character who authenticated by SSO.
- **token_type** (*CharField*) – Token type. The applicable range of the token.
- **character_owner_hash** (*CharField*) – Character owner hash. The unique string identifying this character and its owning EVE account. Changes if the owning account changes.
- **sso_version** (*IntegerField*) – Sso version. EVE SSO Version.

Relationship fields:

Parameters

- **user** (ForeignKey to User) – User. The user to whom this token belongs. (related name: token)
- **scopes** (ManyToManyField to Scope) – Scopes. The access scopes granted by this token. (related name: token)

Reverse relationships:

Parameters
callbackredirect (Reverse ForeignKey from CallbackRedirect) – All callback redirects of this token (related name of token)

property can_refresh: bool

Determine if this token can be refreshed upon expiry.

property expired: bool

Determines if this token has expired.

property expires: datetime

Determines when this token expires.

Returns
Date & time when this token expires

get_esi_client(***kwargs*) → SwaggerClient

Creates an authenticated ESI client with this token.

Parameters
****kwargs** – Extra spec versioning as per *esi.clients.esi_client_factory*

Returns
New ESI client

classmethod get_token(*character_id: int*, *scopes: list*) → *Token*

Helper method to get a token for a specific character with specific scopes.

Parameters

- **character_id** – Character to filter on.
- **scopes** – array of ESI scope strings to search for.

Returns
Matching token or *False* when token is not found

refresh(*session: OAuth2Session = None*, *auth: HTTPBasicAuth = None*) → None

Refresh this token.

Parameters

- **session** – session for refreshing token with
- **auth** – ESI authentication

refresh_or_delete()

Refresh this token or delete it if it can not be refreshed.

valid_access_token() → str

Refresh and return access token to be used in an authed ESI call.

Example

```
# fetch medals for a character
medals = esi.client.Character.get_characters_character_id_medals(
    # required parameter for endpoint
    character_id = token.character_id,
    # provide a valid access token, which will be refreshed if required
    token = token.valid_access_token()
).results()
```

Returns
Valid access token

Raises
TokenExpiredError – When token can not be refreshed

3.5 managers

class TokenQueryset(*model=None*, *query=None*, *using=None*, *hints=None*)

bulk_refresh() → QuerySet

Refresh all refreshable tokens in the queryset and delete any expired token that fails to refresh or can not be refreshed.

Excludes tokens for which the refresh was incomplete for other reasons.

Returns
All refreshed tokens

equivalent_to(*token*) → QuerySet

Fetch all tokens which match the character and scopes of given reference token

Parameters
token – *esi.models.Token* reference token

get_expired() → QuerySet

Get all tokens which have expired.

Returns
All expired tokens.

require_scopes(*scope_string: str | list*) → QuerySet

Filter tokens which have at least a subset of given scopes.

Parameters
scope_string – The required scopes.

Returns
Tokens which have all requested scopes.

require_scopes_exact(*scope_string: str | list*) → QuerySet

Filter tokens which exactly have the given scopes.

Parameters
scope_string – The required scopes.

Returns
Tokens which have all requested scopes.

`require_valid()` → QuerySet

Ensure all tokens are still valid and attempt to refresh any which are expired

Deletes those which fail to refresh or cannot be refreshed.

Returns
All tokens which are still valid.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

esi.app_settings, 5
esi.decorators, 18
esi.errors, 19

INDEX

B

bulk_refresh() (*TokenQueryset method*), 21

C

CachingHttpFuture (*class in esi.clients*), 17
can_refresh (*Token property*), 20

D

DjangoEsiException, 19

E

equivalent_to() (*TokenQueryset method*), 21
esi.app_settings
    module, 5
esi.decorators
    module, 18
esi.errors
    module, 19
ESI_ALWAYS_CREATE_TOKEN (*in module esi.app_settings*), 5
ESI_API_DATASOURCE (*in module esi.app_settings*), 5
ESI_API_VERSION (*in module esi.app_settings*), 5
ESI_CACHE_RESPONSE (*in module esi.app_settings*), 5
esi_client_factory() (*in module esi.clients*), 18
ESI_CONNECTION_ERROR_MAX_RETRIES (*in module esi.app_settings*), 5
ESI_CONNECTION_POOL_MAXSIZE (*in module esi.app_settings*), 5
ESI_DEBUG_RESPONSE_CONTENT_LOGGING (*in module esi.app_settings*), 5
ESI_INFO_LOGGING_ENABLED (*in module esi.app_settings*), 5
ESI_LOG_LEVEL_LIBRARIES (*in module esi.app_settings*), 5
ESI_REQUESTS_CONNECT_TIMEOUT (*in module esi.app_settings*), 5
ESI_REQUESTS_READ_TIMEOUT (*in module esi.app_settings*), 5
ESI_SERVER_ERROR_BACKOFF_FACTOR (*in module esi.app_settings*), 5
ESI_SERVER_ERROR_MAX_RETRIES (*in module esi.app_settings*), 5
ESI_USER_CONTACT_EMAIL (*in module esi.app_settings*), 5
EsiClientProvider (*class in esi.clients*), 18
expired (*Token property*), 20
expires (*Token property*), 20

G

get_esi_client() (*Token method*), 20
get_expired() (*TokenQueryset method*), 21
get_token() (*Token class method*), 20

I

IncompleteResponseError, 19

M

module
    esi.app_settings, 5
    esi.decorators, 18
    esi.errors, 19

N

NotRefreshableTokenError, 19

R

refresh() (*Token method*), 20
refresh_or_delete() (*Token method*), 20
require_scopes() (*TokenQueryset method*), 21
require_scopes_exact() (*TokenQueryset method*), 21
require_valid() (*TokenQueryset method*), 22
result() (*CachingHttpFuture method*), 17
results() (*CachingHttpFuture method*), 17
results_localized() (*CachingHttpFuture method*), 17

S

single_use_token() (*in module esi.decorators*), 18

T

Token (*class in esi.models*), 19
token_required() (*in module esi.decorators*), 18
TokenError, 19

TokenExpiredError, 19
TokenInvalidError, 19
TokenQueryset (*class in esi.managers*), 21
tokens_required() (*in module esi.decorators*), 18

V

valid_access_token() (*Token method*), 20