OpenReview Python API Documentation

Release 0.0.1

OpenReview Team

Feb 13, 2019

Contents

CHAPTER 1

About OpenReview

OpenReview aims to promote openness in scientific communication, particularly the peer review process, by providing a flexible cloud-based web interface and underlying database API.

This document is a guide to the python API supported by OpenReview.

CHAPTER 2

Contents

2.1 Read more about OpenReview

OpenReview aims to promote openness in scientific communication, particularly the peer review process, by providing a flexible cloud-based web interface and underlying database API enabling the following:

Open Peer Review

We provide a configurable platform for peer review that generalizes over many subtle gradations of openness, allowing conference organizers, journals, and other "reviewing entities" to configure the specific policy of their choice. We intend to act as a testbed for different policies, to help scientific communities experiment with open scholarship while addressing legitimate concerns regarding confidentiality, attribution, and bias.

Open Publishing

Track submissions, coordinate the efforts of editors, reviewers and authors, and host... Sharded and distributed for speed and reliability.

Open Access

Free access to papers for all, free paper submissions. No fees.

Open Discussion

Hosting of accepted papers, with their reviews, comments. Continued discussion forum associated with the paper post acceptance. Publication venue chairs/editors can control structure of review/comment forms, read/write access, and its timing.

Open Directory

Collection of people, with conflict-of-interest information, including institutions and relations, such as co-authors, co-PIs, co-workers, advisors/advisees, and family connections.

Open Recommendations

Models of scientific topics and expertise. Directory of people includes scientific expertise. Reviewer-paper matching for conferences with thousands of submissions, incorporating expertise, bidding, constraints, and reviewer balancing of various sorts. Paper recommendation to users.

Open API

We provide a simple REST API for accessing and uploading records of people, their groupings, document content, invitations and reviewing assignments, conflict-of-interest designations, and reviewing workflow patterns. You can then write scripts , all with a clear, robust model of read/write permissions. Track submissions, monitor review process, send customized bulk email messages, automate workflow actions.

Open Source

We are committed to open source. Look for our public repo on GitHub soon.

2.2 API Documentation

2.2.1 Client

class openreview.**Client**(*baseurl=None*, *username=None*, *password=None*, *token=None*)

activate_user(*token*, *content*)
Activates a newly registered user

Parameters

- **token** – activation token
- **content** – content of the profile to activate

Example Usage: >>> res = client.activate_user('new@user.com', {

'names': [

{ 'first': 'New', 'last': 'User', 'username': '~New_User1'

}

],

'emails': ['new@user.com'], 'preferredEmail': 'new@user.com' })

add_members_to_group(*group*, *members*)

Adds members to a group
Members should be in a string, unicode or a list format

delete_note(*note*)
Deletes the note and returns a {status = 'ok'} in case of a successful deletion and an OpenReview exception otherwise.

get_activatable(*token=None*)
Returns the activation token for a registered user

get_group(*id*)
Returns a single Group by id if available

get_groups(*id=None*, *regex=None*, *member=None*, *host=None*, *signatory=None*, *limit=None*, *offset=None*)
Returns a list of Group objects based on the filters provided.

get_invitation(*id*)
Returns a single invitation by id if available

get_invitations(*id=None*, *invitee=None*, *replytoNote=None*, *replyForum=None*, *signature=None*, *note=None*, *regex=None*, *tags=None*, *limit=None*, *offset=None*, *mindue-date=None*, *duedate=None*, *pastdue=None*, *replyto=None*, *details=None*)
Returns a list of Invitation objects based on the filters provided.

get_note(*id*)
Returns a single note by id if available

get_notes(*id=None*, *paperhash=None*, *forum=None*, *invitation=None*, *replyto=None*, *tauthor=None*, *signature=None*, *writer=None*, *trash=None*, *number=None*, *content=None*, *limit=None*, *offset=None*, *mintcdate=None*, *details=None*)
Returns a list of Note objects based on the filters provided.

Parameters

- **id** – a Note ID. If provided, returns Notes whose ID matches the given ID.
- **paperhash** – a "paperhash" for a note. If provided, returns Notes whose paperhash matches this argument. (A paperhash is a human-interpretable string built from the Note's title and list of authors to uniquely identify the Note)
- **forum** – a Note ID. If provided, returns Notes whose forum matches the given ID.
- **invitation** – an Invitation ID. If provided, returns Notes whose "invitation" field is this Invitation ID.
- **replyto** – a Note ID. If provided, returns Notes whose replyto field matches the given ID.
- **tauthor** – a Group ID. If provided, returns Notes whose tauthor field ("true author") matches the given ID, or is a transitive member of the Group represented by the given ID.
- **signature** – a Group ID. If provided, returns Notes whose signatures field contains the given Group ID.
- **writer** – a Group ID. If provided, returns Notes whose writers field contains the given Group ID.
- **trash** – a Boolean. If True, includes Notes that have been deleted (i.e. the ddate field is less than the current date)
- **number** – an integer. If present, includes Notes whose number field equals the given integer.
- **content** – a dictionary. If present, includes Notes whose each key is present in the content field and it is equals the given value.
- **mintcdate** – an integer representing an Epoch time timestamp, in milliseconds. If provided, returns Notes whose "true creation date" (tcdate) is at least equal to the value of mintcdate.
- **details** – TODO: What is a valid value for this field?

get_pdf(*id*)
Returns the binary content of a pdf using the provided note id If the pdf is not found then this returns an error message with "status":404

Example Usage:

```
>>> f = get_pdf(id='Place Note-ID here')
>>> with open('output.pdf','wb') as op: op.write(f)
```

get_profile(*email_or_id*)
Returns a single profile (a note) by id, if available

get_profiles(*email_or_id_list=None*, *id=None*, *email=None*, *first=None*, *middle=None*, *last=None*)

If the list is tilde_ids, returns an array of profiles

If the list is emails, returns an array of dictionaries with 'email' and 'profile'

get_references(*referent=None*, *invitation=None*, *mintcdate=None*, *limit=None*, *offset=None*, *original=False*)

Returns a list of revisions for a note.

Parameters

- **referent** – a Note ID. If provided, returns references whose "referent" value is this Note ID.
- **invitation** – an Invitation ID. If provided, returns references whose "invitation" field is this Invitation ID.
- **mintcdate** – an integer representing an Epoch time timestamp, in milliseconds. If provided, returns references whose "true creation date" (tcdate) is at least equal to the value of mintcdate.
- **original** – a boolean. If True then get_references will additionally return the references to the original note.

get_tag(*id*)

Returns a single tag by id if available

get_tags(*id=None*, *invitation=None*, *forum=None*, *limit=None*, *offset=None*)

Returns a list of Tag objects based on the filters provided.

Parameters

- **id** – a Tag ID. If provided, returns Tags whose ID matches the given ID.
- **forum** – a Note ID. If provided, returns Tags whose forum matches the given ID.
- **invitation** – an Invitation ID. If provided, returns Tags whose "invitation" field is this Invitation ID.

get_tildeusername(*first*, *last*, *middle=None*)

Returns next possible tilde user name corresponding to the specified first, middle and last name

First and last names are required, while middle name is optional

login_user(*username=None*, *password=None*)

Logs in a registered user and returns authentication token

post_group(*group*, *overwrite=True*)

Posts the group. Upon success, returns the posted Group object.

If the group is unsigned, signs it using the client's default signature.

post_invitation(*invitation*)

Posts the invitation. Upon success, returns the posted Invitation object.

If the invitation is unsigned, signs it using the client's default signature.

post_note(*note*)

Posts the note. Upon success, returns the posted Note object.

If the note is unsigned, signs it using the client's default signature

post_profile(*profile*)
Posts the profile

post_tag(*tag*)
Posts the tag. Upon success, returns the posted Tag object.

put_pdf(*fname*)
Uploads a pdf to the openreview server and returns a relative url for the uploaded pdf

Parameters **fname** – path to the pdf

register_user(*email=None*, *first=None*, *last=None*, *middle=''*, *password=None*)
Registers a new user

remove_members_from_group(*group*, *members*)

Removes members from a group

Members should be in a string, unicode or a list format

search_notes(*term*, *content='all'*, *group='all'*, *source='all'*, *limit=None*, *offset=None*)
Searches notes based on term, content, group and source as the criteria

send_mail(*subject*, *recipients*, *message*)
Sends emails to a list of recipients

update_profile(*profile*)
Updates the profile

class openreview.**Group**(*id*, *readers*, *writers*, *signatories*, *signatures*, *cdate=None*, *ddate=None*, *members=None*, *nonreaders=None*, *web=None*)

add_member(*member*)
Adds a member to the group

add_webfield(*web*)
Adds a webfield to the group

classmethod from_json(*g*)
Returns a deserialized object from a json string

Parameters **g** – The json string consisting of a serialized object of type "Group"

post(*client*)
Posts a group

remove_member(*member*)
Removes a member from the group

to_json()
Returns serialized json string for a given object

class openreview.**Invitation**(*id*, *readers=None*, *writers=None*, *invitees=None*, *signatures=None*, *reply=None*, *super=None*, *noninvitees=None*, *nonreaders=None*, *web=None*, *process=None*, *process_string=None*, *duedate=None*, *expdate=None*, *cdate=None*, *rdate=None*, *ddate=None*, *tcdate=None*, *tmdate=None*, *multiReply=None*, *taskCompletionCount=None*, *transform=None*, *details=None*)

classmethod from_json(*i*)
Returns a deserialized object from a json string

Parameters **i** – The json string consisting of a serialized object of type "Invitation"

to_json()
Returns serialized json string for a given object

class openreview.**Note**(*invitation*, *readers*, *writers*, *signatures*, *content*, *id=None*, *original=None*, *number=None*, *cdate=None*, *tcdate=None*, *tmdate=None*, *ddate=None*, *forum=None*, *referent=None*, *replyto=None*, *nonreaders=None*, *details=None*, *tauthor=None*)

classmethod from_json(*n*)
Returns a deserialized object from a json string

Parameters **n** – The json string consisting of a serialized object of type "Note"

to_json()
Returns serialized json string for a given object

class openreview.**Tag**(*tag*, *invitation*, *readers*, *signatures*, *id=None*, *cdate=None*, *tcdate=None*, *ddate=None*, *forum=None*, *replyto=None*, *nonreaders=None*)

classmethod from_json(*t*)
Returns a deserialized object from a json string

Parameters **t** – The json string consisting of a serialized object of type "Tag"

to_json()
Returns serialized json string for a given object

class openreview.**Profile**(*id=None*, *active=None*, *password=None*, *number=None*, *tcdate=None*, *tmdate=None*, *referent=None*, *packaging=None*, *invitation=None*, *readers=None*, *nonreaders=None*, *signatures=None*, *writers=None*, *content=None*, *metaContent=None*, *tauthor=None*)

2.2.2 Tools

tools.**add_assignment**(*client*, *paper_number*, *conference*, *reviewer*, *parent_group_params={}*, *individual_group_params={}*, *parent_label='Reviewers'*, *individual_label='AnonReviewer'*, *use_profile=True*)

Assigns a reviewer to a paper.

Also adds the given user to the parent and individual groups defined by the paper number, conference, and labels

"individual groups" are groups with a single member;

e.g. conference.org/Paper1/AnonReviewer1

"parent group" is the group that contains the individual groups;

e.g. conference.org/Paper1/Reviewers

Parameters

- **paper_number** – the number of the paper to assign
- **conference** – the ID of the conference being assigned
- **reviewer** – may be an email address or a tilde ID;
- **parent_group_params** – optional parameter that overrides the default

- **individual_group_params** – optional parameter that overrides the default

tools.**assign**(*client*, *paper_number*, *conference*, *parent_group_params={}*, *individual_group_params={}*, *reviewer_to_add=None*, *reviewer_to_remove=None*, *parent_label='Reviewers'*, *individual_label='AnonReviewer'*, *use_profile=True*)

DEPRECATED as of openreview-py revision 0.9.5

Either assigns or unassigns a reviewer to a paper. TODO: Is this function really necessary?

"individual groups" are groups with a single member; e.g. conference.org/Paper1/AnonReviewer1 "parent group" is the group that contains the individual groups; e.g. conference.org/Paper1/Reviewers

Parameters

- **paper_number** – the number of the paper to assign
- **conference** – the ID of the conference being assigned
- **parent_group_params** – optional parameter that overrides the default
- **individual_group_params** – optional parameter that overrides the default
- **reviewer_to_add** – may be an email address or a tilde ID; adds the given user to the parent and individual groups defined by the paper number, conference, and labels
- **reviewer_to_remove** – same as @reviewer_to_add, but removes the user

It's important to remove any users first, so that we can do direct replacement of one user with another.

For example: passing in a reviewer to remove AND a reviewer to add should replace the first user with the second.

tools.**build_groups**(*conference_group_id*, *default_params=None*)

Given a group ID, returns a list of empty groups that correspond to the given group's subpaths

(e.g. Test.com, Test.com/TestConference, Test.com/TestConference/2018)

```
>>> [group.id for group in build_groups('ICML.cc/2019/Conference')]
[u'ICML.cc', u'ICML.cc/2019', u'ICML.cc/2019/Conference']
```

tools.**create_profile**(*client*, *email*, *first*, *last*, *middle=None*, *allow_duplicates=False*)

Given email, first name, last name, and middle name (optional), creates and returns a user profile.

If a profile with the same name exists, and allow_duplicates is False, an exception is raised.

If a profile with the same name exists and allow_duplicates is True, a profile is created with the next largest number (e.g. if ~Michael_Spector1 exists, ~Michael_Spector2 will be created)

tools.**datetime_millis**(*dt*)

Convets a datetim to milliseconds.

tools.**fill_template**(*template*, *paper*)

Fills an openreview "template" with the corresponding values from an openreview.Note object. Templates are dicts that match the schema of any OpenReview object class .

Example: group_template = {

'id': 'Conf.org/2019/Paper<number>', 'members': ['Conf.org/2019/Paper<number>/Reviewers']

}

Parameters **template** – a dict that matches the schema of an OpenReview *Group*

or *Invitation* with any number of wildcards in the form of "<attr>", where "attr" is an attribute in the *Note* class. :arg paper: an instance of *Note* class, to fill the template values.

tools.**get_all_venues**(*client*)
Returns a list of all the venues

Parameters **client** – Object of *Client* class

tools.**get_bibtex**(*note*, *venue_fullname*, *year*, *url_forum=None*, *accepted=False*, *anonymous=True*)
Generates a bibtex field for a given Note.

The "accepted" argument is used to indicate whether or not the paper was ultimately accepted. (OpenReview generates bibtex fields for rejected papers)

The "anonymous" argument is used to indicate whether or not the paper's authors should be revealed.

Warning: this function is a work-in-progress.

tools.**get_paper_conflicts**(*client*, *paper*)
Given a Note object representing a submitted paper, returns a tuple containing two sets: domain_conflicts and relation_conflicts.

domain_conflicts is a set of domains/subdomains that may have a conflict of interest with the given paper.

relation_conflicts is a set of group IDs (email addresses or profiles) that may have a conflict of interest with the given paper.

Automatically ignores domain conflicts with "gmail.com".

tools.**get_paperhash**(*first_author*, *title*)
Returns the paperhash of a paper, given the title and first author.

```
>>> get_paperhash('David Soergel', 'Open Scholarship and Peer Review: a Time for␣
↪Experimentation')
u'soergel|open_scholarship_and_peer_review_a_time_for_experimentation'
```

tools.**get_preferred_name**(*profile*)
Returns a string representing the user's preferred name, if available, or the first listed name if not available.

Accepts openreview.Profile object

tools.**get_profile_conflicts**(*client*, *reviewer_to_add*)
Helper function for profile_conflicts function. Given a reviewer ID or email address, requests the server for that reviewer's profile, and checks it for conflicts using profile_conflicts.

tools.**get_reviewer_groups**(*client*, *paper_number*, *conference*, *group_params*, *parent_label*, *individual_label*)
This is only intended to be used as a local helper function

Parameters

- **paper_number** – the number of the paper to assign
- **conference** – the ID of the conference being assigned
- **group_params** – optional parameter that overrides the default

tools.**get_submission_invitations**(*client*, *open_only=False*)
Returns a list of invitation ids visible to the client according to the value of parameter "open_only".

Parameters

- **client** – Object of *Client* class
- **open_only** – Default value is False. This is a boolean param with value True implying that the results would be invitations having a future due date.

Example Usage:

```
>>> get_submission_invitations(c,True)
[u'machineintelligence.cc/MIC/2018/Conference/-/Submission', u
↪'machineintelligence.cc/MIC/2018/Abstract/-/Submission', u'ISMIR.net/2018/WoRMS/
↪-/Submission', u'OpenReview.net/Anonymous_Preprint/-/Submission']
```

tools.**iterget_groups**(*client*, *id=None*, *regex=None*, *member=None*, *host=None*, *signatory=None*)

Returns an iterator over groups, filtered by the provided parameters, ignoring API limit.

Parameters

- **client** – an openreview.Client object.
- **id** – a Note ID. If provided, returns groups whose "id" value is this Group ID.
- **regex** – a regular expression string to match Group IDs. If provided, returns groups whose "id" value matches the given regex.
- **member** – a string. Essentially, members field contains Group Ids that are members of this Group object. If provided, returns groups whose "members" field contains the given string.

tools.**iterget_invitations**(*client*, *id=None*, *invitee=None*, *regex=None*, *tags=None*, *mindue-date=None*, *duedate=None*, *pastdue=None*, *replytoNote=None*, *re-plyForum=None*, *signature=None*, *note=None*, *replyto=None*, *de-tails=None*)

Returns an iterator over invitations, filtered by the provided parameters, ignoring API limit.

Parameters

- **client** – an openreview.Client object.
- **id** – an Invitation ID. If provided, returns invitations whose "id" value is this Invitation ID.
- **invitee** – a string. Essentially, invitees field in an Invitation object contains Group Ids being invited using the invitation. If provided, returns invitations whose "invitee" field contains the given string.
- **regex** – a regular expression string to match Invitation IDs. If provided, returns invitations whose "id" value matches the given regex.

tools.**iterget_notes**(*client*, *id=None*, *paperhash=None*, *forum=None*, *invitation=None*, *re-plyto=None*, *tauthor=None*, *signature=None*, *writer=None*, *trash=None*, *number=None*, *mintcdate=None*, *content=None*, *details=None*)

Returns an iterator over Notes, filtered by the provided parameters, ignoring API limit.

Parameters

- **client** – an openreview.Client object.
- **id** – a Note ID. If provided, returns Notes whose ID matches the given ID.
- **paperhash** – a "paperhash" for a note. If provided, returns Notes whose paperhash matches this argument. (A paperhash is a human-interpretable string built from the Note's title and list of authors to uniquely identify the Note)
- **forum** – a Note ID. If provided, returns Notes whose forum matches the given ID.
- **invitation** – an Invitation ID. If provided, returns Notes whose "invitation" field is this Invitation ID.
- **replyto** – a Note ID. If provided, returns Notes whose replyto field matches the given ID.
- **tauthor** – a Group ID. If provided, returns Notes whose tauthor field ("true author") matches the given ID, or is a transitive member of the Group represented by the given ID.

- **signature** – a Group ID. If provided, returns Notes whose signatures field contains the given Group ID.
- **writer** – a Group ID. If provided, returns Notes whose writers field contains the given Group ID.
- **trash** – a Boolean. If True, includes Notes that have been deleted (i.e. the ddate field is less than the current date)
- **number** – an integer. If present, includes Notes whose number field equals the given integer.
- **mintcdate** – an integer representing an Epoch time timestamp, in milliseconds. If provided, returns Notes whose "true creation date" (tcdate) is at least equal to the value of mintcdate.
- **content** – a dictionary. If present, includes Notes whose each key is present in the content field and it is equals the given value.
- **details** – TODO: What is a valid value for this field?

tools.**iterget_references**(*client*, *referent=None*, *invitation=None*, *mintcdate=None*)

Returns an iterator over references, filtered by the provided parameters, ignoring API limit.

Parameters

- **client** – an openreview.Client object.
- **referent** – a Note ID. If provided, returns references whose "referent" value is this Note ID.
- **invitation** – an Invitation ID. If provided, returns references whose "invitation" field is this Invitation ID.
- **mintcdate** – an integer representing an Epoch time timestamp, in milliseconds. If provided, returns references whose "true creation date" (tcdate) is at least equal to the value of mintcdate.

tools.**iterget_tags**(*client*, *id=None*, *invitation=None*, *forum=None*)

Returns an iterator over Tags, filtered by the provided parameters, ignoring API limit.

Example: iterget_tags(client, invitation='MyConference.org/-/Bid_Tags')

Parameters

- **id** – a Tag ID. If provided, returns Tags whose ID matches the given ID.
- **forum** – a Note ID. If provided, returns Tags whose forum matches the given ID.
- **invitation** – an Invitation ID. If provided, returns Tags whose "invitation" field is this Invitation ID.

tools.**next_individual_suffix**(*unassigned_individual_groups*, *individual_groups*, *individual_label*)

"individual groups" are groups with a single member;
e.g. conference.org/Paper1/AnonReviewer1

Parameters

- **unassigned_individual_groups** – a list of individual groups with no members
- **individual_groups** – the full list of individual groups, empty or not
- **individual_label** – the "label" of the group: e.g. "AnonReviewer"

Returns

an individual group's suffix (e.g. AnonReviewer1)

The suffix will be the next available empty group,

or will be the suffix of the largest indexed group +1

tools.**post_group_parents**(*client*, *group*, *overwrite_parents=False*)
Helper function for posting groups created by build_groups function.

Recommended that this function be deprecated.

tools.**post_submission_groups**(*client*, *conference_id*, *submission_invite*, *chairs*)
Create paper group, authors group, reviewers group, review non-readers group for all notes returned by the submission_invite.

tools.**profile_conflicts**(*profile*)
Given a profile, returns a tuple containing two sets: domain_conflicts and relation_conflicts.

domain_conflicts is a set of domains/subdomains that may have a conflict of interest with the given profile.

relation_conflicts is a set of group IDs (email addresses or profiles) that may have a conflict of interest with the given profile.

tools.**recruit_reviewer**(*client*, *user*, *first*, *hash_seed*, *recruit_reviewers_id*, *recruit_message*, *recruit_message_subj*, *reviewers_invited_id*, *verbose=True*)
Recruit a reviewer. Sends an email to the reviewer with a link to accept or reject the recruitment invitation.

Parameters

- **hash_seed** – a random number for seeding the hash.
- **recruit_message** – a formattable string containing the following string variables: (name, accept_url, decline_url)
- **recruit_message_subj** – subject line for the recruitment email
- **reviewers_invited_id** – group ID for the "Reviewers Invited" group, often used to keep track of which reviewers have already been emailed.

tools.**remove_assignment**(*client*, *paper_number*, *conference*, *reviewer*, *parent_group_params={}*, *parent_label='Reviewers'*, *individual_label='AnonReviewer'*)

Un-assigns a reviewer from a paper.
Removes the given user from the parent group, and any assigned individual groups.

"individual groups" are groups with a single member;
e.g. conference.org/Paper1/AnonReviewer1
"parent group" is the group that contains the individual groups;
e.g. conference.org/Paper1/Reviewers

Parameters

- **paper_number** – the number of the paper to assign
- **conference** – the ID of the conference being assigned
- **reviewer** – same as @reviewer_to_add, but removes the user
- **parent_group_params** – optional parameter that overrides the default

- **individual_group_params** – optional parameter that overrides the default

tools.**replace_members_with_ids**(*client*, *group*)
Given a Group object, iterates through the Group's members and, for any member represented by an email address, attempts to find a profile associated with that email address. If a profile is found, replaces the email with the profile ID.

Returns None.

tools.**subdomains**(*domain*)
Given an email address, returns a list with the domains and subdomains.

```
>>> subdomains('johnsmith@iesl.cs.umass.edu')
[u'iesl.cs.umass.edu', u'cs.umass.edu', u'umass.edu']
```

tools.**timestamp_GMT**(*year*, *month*, *day*, *hour=0*, *minute=0*, *second=0*)
Given year, month, day, and (optionally) hour, minute, second in GMT time zone: returns the number of milliseconds between this day and Epoch Time (Jan 1, 1970).

```
>>> timestamp_GMT(1990, 12, 20, hour=12, minute=30, second=24)
661696224000
```

2.3 Installation and Setup

2.3.1 Installation

Install the `openreview-py` package with pip:

```
pip install openreview-py
```

2.3.2 Setup

Once installation is done, this python package can be used to make api calls to perform several operations (as listed in API documentation)

To access Openreview API resources, an object of the Client class is needed.:

```
>>> client = openreview.Client(baseurl='https://openreview.net',username=<your␣
↪username>,password=<your password>)
```

While a logged in user gets all the Openreview services, some of the services can be accessed by a guest (without logging in).:

```
>>> guest_client = openreview.Client(baseurl='https://openreview.net')
```

2.3.3 Run various requests in parallel

Use a multiprocessing pool to run a set of requests in parallel

```
>>> notes = list(tools.iterget_notes(client, invitation='ICLR.cc/2019/Conference/-/
↪Blind_Submission'))
>>> openreview.tools.parallel_exec(notes, do_work)
```

The function do_work has to be defined as pickled function, more info: https://docs.python.org/3/library/pickle.html#what-can-be-pickled-and-unpickled

2.4 Examples

2.4.1 Creating a conference

Demo Workshop

Here we explain some of the steps for creation of a new conference on OpenReview system.

Some of the tasks described below (such as creation of a conference sub-group, creation of a submission, etc) can only be done using an administrator profile.

If you are registered as an administrator with OpenReview, you should already have admin privileges on a group representing your conference (e.g. ICML.cc/2019/Conference).

Following are the steps and brief explanations.

Login

Login can be done through the API

```
>>> import openreview
>>> client = openreview.Client(baseurl='https://openreview.net',username=<your␣
↪username>,password=<your password>)
```

Creating a conference

You need admin privileges for creating subgroups within a group representing your conference (e.g. ICML.cc). When you create new groups, they must be subgroups of this conference (e.g. ICML.cc/2019)"

To create the conference you represent, Openreview team will create a group

```
>>> client.post_group(openreview.Group(id = 'ICML.cc',
                                       readers = ['everyone'],
                                       writers = ['OpenReview.net'],
                                       signatories = ['ICML.cc'],
                                       signatures = ['OpenReview.net']))
```

```
{'cdate': 1532031329209,
'ddate': None,
'id': u'ICML.cc',
'members': [],
'nonreaders': [],
'readers': [u'everyone'],
'signatories': [u'ICML.cc'],
'signatures': [u'OpenReview.net'],
'web': None,
'writers': [u'OpenReview.net']}
```

Once you (the conference admin) receive information from the Openreview team that your conference root has been setup, you can begin to create subgroups under the root. For instance, to create an ICML.cc conference for 2019 following steps should be done by the conference admin:

```
>>> client.post_group(openreview.Group(id = 'ICML.cc/2019',
                                       readers = ['everyone'],
                                       writers = ['OpenReview.net'],
                                       signatories = ['ICML.cc/2019'],
                                       signatures = ['OpenReview.net']))
```

```
{'cdate': 1532031902863,
'ddate': None,
'id': u'ICML.cc/2019',
'members': [],
'nonreaders': [],
'readers': [u'everyone'],
'signatories': [u'ICML.cc/2019'],
'signatures': [u'ICML.cc'],
'web': None,
'writers': [u'ICML.cc']}
```

Next step would be to create another subgroup 'ICML.cc/2019/Conference':

```
>>> client.post_group(openreview.Group(id = 'ICML.cc/2019/Conference',
                                       readers = ['everyone'],
                                       writers = ['ICML.cc/2019'],
                                       signatories = ['ICML.cc/2019/Conference'],
                                       signatures = ['ICML.cc/2019'],
                                       members = [<Tilde id of members>],
                                       web = <Absolute path to JS file>))
```

```
{'cdate': 1531339441440,
'ddate': None,
'id': u'ICML.cc/2019/Conference',
'members': [],
'nonreaders': [],
'readers': [u'everyone'],
'signatories': [u'ICML.cc/2019/Conference'],
'signatures': [u'ICML.cc/2019'],
'web': u'\n// ---------------------------------\n// Basic venue homepage␣
↪template\n//\n// This webfield displays the conference header (#header), the submit␣
↪button (#invitation),\n// and a list of all submitted papers (#notes).\n// ---------
↪---------------------------\n\n// Constants\nvar CONFERENCE = "ICML.cc/2019/
↪Conference";\nvar INVITATION = CONFERENCE + \'/-/Submission\';\nvar SUBJECT_AREAS =␣
↪[\n  // Add conference specific subject areas here\n];\nvar BUFFER = 1000 * 60 * 30;
↪  // 30 minutes\nvar PAGE_SIZE = 50;\n\nvar paperDisplayOptions = {\n  pdfLink:␣
↪true,\n  replyCount: true,\n  showContents: true\n};\n\n// Main is the entry point␣
↪to the webfield code and runs everything\nfunction main() {\n  Webfield.ui.setup(\'
↪#group-container\', CONFERENCE);  // required\n\n  renderConferenceHeader();\n\n␣
↪load().then(render).then(function() {\n    Webfield.setupAutoLoading(INVITATION,␣
↪PAGE_SIZE, paperDisplayOptions);\n  });\n}\n\n// RenderConferenceHeader renders the␣
↪static info at the top of the page. Since that content\n// never changes, put it in␣
↪its own function\nfunction renderConferenceHeader() {\n  Webfield.ui.venueHeader(
↪{\n    title: "ICML ",\n    subtitle: "Recent Advances in Ubiquitous Computing",\n␣
↪  location: "University of Rostock, Germany",\n    date: "2017, August 04",\n  ␣
↪website: "https://studip.uni-rostock.de/seminar_main.php?
↪auswahl=c9b2fd0a6f525ce968d41d737de3ccb5",\n    instructions: null,  // Add any␣
↪custom instructions here. Accepts HTML\n    deadline: "Submission Deadline: 2017,␣
↪June 15th at 11:59 pm (CEST) "\n  });\n\n  Webfield.ui.spinner(\'#notes\');\n}\n\n//
↪ Load makes all the API calls needed to get the data to render the page\n// It␣
↪returns a jQuery deferred object: https://api.jquery.com/category/deferred-object/
↪\nfunction load() {\n  var invitationP = Webfield.api.
↪getSubmissionInvitation(INVITATION, {deadlineBuffer: BUFFER});\n  var notesP =␣
↪Webfield.api.getSubmissions(INVITATION, {pageSize: PAGE_SIZE});\n\n  return $.
↪when(invitationP, notesP);\n}\n\n// Render is called when all the data is finished␣
↪being loaded from the server\n// It should also be called when the page needs to be␣
↪refreshed, for example after a user\n// submits a new paper.\nfunction␣
↪render(invitation, notes) {\n  // Display submission button and form\n  $(\'
```

(continues on next page)

(continued from previous page)

```
'writers': [u'ICML.cc/2019']}
```

Please note that this conference group does not show up under the header "Open for Submissions" on Openreview homepage unless an invitation for submission with a future due date is created (as shown in the screenshot below).

Although, the conference 'ICML.cc/2019/Conference' that we just created is not accessible from "Open for Submissions", it can still be found on the Openreview homepage under the header "All Venues".

Note that the conference's URL is essentially the openreview url with the name of the conference group, i.e. "https://openreview.net/group?id=ICML.cc/2019/Conference".

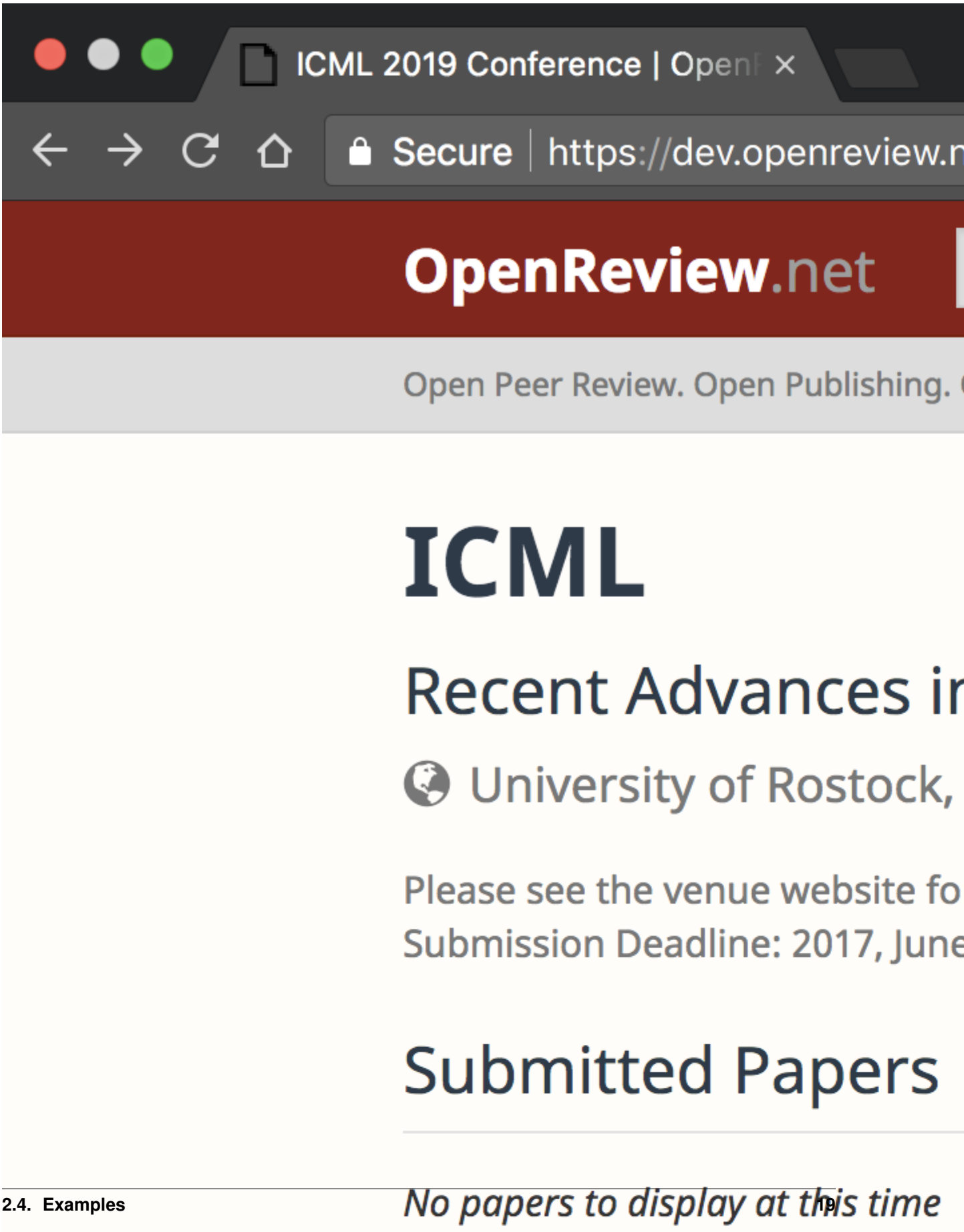
Note that the web field for the conference is either JS code or the absolute path of a JS file containing the code.

Sample JS file

```
// -------------------------------------
// Basic venue homepage template
//
// This webfield displays the conference header (#header), the submit button (
↪#invitation),
// and a list of all submitted papers (#notes).
// -------------------------------------

// Constants
var CONFERENCE = "ICML.cc/2019/Conference";
var INVITATION = CONFERENCE + '/-/Submission';
var SUBJECT_AREAS = [
  // Add conference specific subject areas here
];
var BUFFER = 1000 * 60 * 30;  // 30 minutes
var PAGE_SIZE = 50;

var paperDisplayOptions = {
  pdfLink: true,
  replyCount: true,
  showContents: true
};

// Main is the entry point to the webfield code and runs everything
function main() {
  Webfield.ui.setup('#group-container', CONFERENCE);  // required

  renderConferenceHeader();

  load().then(render).then(function() {
    Webfield.setupAutoLoading(INVITATION, PAGE_SIZE, paperDisplayOptions);
  });
}

// RenderConferenceHeader renders the static info at the top of the page.
function renderConferenceHeader() {
  Webfield.ui.venueHeader({
    title: "ICML ",
    subtitle: "Recent Advances in Ubiquitous Computing",
    location: "University of Rostock, Germany",
    date: "2017, August 04",
```

(continues on next page)


Venues | OpenReview
Secure
https://dev.openreview.n
OpenReview.net
Open Peer Review. Open Publishing.
Open for Sub
ICLR 2018 Conf
All Venues
ICLR 2018
MIDL 2018


ICML 2019 Conference | Open
Secure
https://dev.openreview.n
OpenReview.net
Open Peer Review. Open Publishing.
ICML
Recent Advances ir
University of Rostock,
Please see the venue website fo
Submission Deadline: 2017, June
Submitted Papers
No papers to display at this time

(continued from previous page)

```
    website: "https://studip.uni-rostock.de/seminar_main.php?
↪auswahl=c9b2fd0a6f525ce968d41d737de3ccb5",
    instructions: null,  // Add any custom instructions here. Accepts HTML
    deadline: "Submission Deadline: 2017, June 15th at 11:59 pm (CEST) "
  });

  Webfield.ui.spinner('#notes');
}

// Load makes all the API calls needed to get the data to render the page
// It returns a jQuery deferred object: https://api.jquery.com/category/deferred-
↪object/
function load() {
  var invitationP = Webfield.api.getSubmissionInvitation(INVITATION, {deadlineBuffer:␣
↪BUFFER});
  var notesP = Webfield.api.getSubmissions(INVITATION, {pageSize: PAGE_SIZE});

  return $.when(invitationP, notesP);
}

// Render is called when all the data is finished being loaded from the server
// It should also be called when the page needs to be refreshed, for example after a␣
↪user
// submits a new paper.
function render(invitation, notes) {
  // Display submission button and form
  $('#invitation').empty();
  Webfield.ui.submissionButton(invitation, user, {
    onNoteCreated: function() {
      // Callback funtion to be run when a paper has successfully been submitted␣
↪(required)
      load().then(render).then(function() {
        Webfield.setupAutoLoading(INVITATION, PAGE_SIZE, paperDisplayOptions);
      });
    }
  });

  // Display the list of all submitted papers
  $('#notes').empty();
  Webfield.ui.submissionList(notes, {
    heading: 'Submitted Papers',
    displayOptions: paperDisplayOptions,
    search: {
      enabled: true,
      subjectAreas: SUBJECT_AREAS,
      onResults: function(searchResults) {
        Webfield.ui.searchResults(searchResults, paperDisplayOptions);
        Webfield.disableAutoLoading();
      },
      onReset: function() {
        Webfield.ui.searchResults(notes, paperDisplayOptions);
        Webfield.setupAutoLoading(INVITATION, PAGE_SIZE, paperDisplayOptions);
      }
    }
  });
}
```

(continues on next page)

(continued from previous page)

```
// Go!
main();
```

Creating Submission Invitations

If you have administrator privileges in OpenReview, you will be able to create Invitations for Submissions for your conference using this API

```
>>> client.post_invitation(openreview.Invitation(id = 'ICML.cc/2019/Conference/-/
↪Submission',
                                        readers = ['everyone'],
                                        writers = ['ICML.cc/2019/Conference'],
                                        signatures = ['ICML.cc/2019/Conference'],
                                        invitees = ['everyone'],
                                        duedate = 1562875092000,
                                        reply = {
                                                'forum': None,
                                                'replyto': None,
                                                'readers': {
                                                    'description': 'The users who␣
↪will be allowed to read the above content.',
                                                    'values': ['everyone']
                                                },
                                                'signatures': {
                                                    'description': 'Your authorized␣
↪identity to be associated with the above content.',
                                                    'values-regex': '~.*'
                                                },
                                                'writers': {
                                                    'values-regex': '~.*'
                                                },
                                                'content':{
                                                    'title': {
                                                        'description': 'Title of␣
↪paper.',
                                                        'order': 1,
                                                        'value-regex': '.{1,250}',
                                                        'required':True
                                                    },
                                                    'authors': {
                                                        'description': 'Comma␣
↪separated list of author names. Please provide real names; identities will be␣
↪anonymized.',
                                                        'order': 2,
                                                        'values-regex': "[^;,\\n]+(,[^
↪,\\n]+)*",
                                                        'required':True
                                                    },
                                                    'authorids': {
                                                        'description': 'Comma␣
↪separated list of author email addresses, lowercased, in the same order as above.␣
↪For authors with existing OpenReview accounts, please make sure that the provided␣
↪email address(es) match those listed in the author\'s profile. Please provide real␣
↪emails; identities will be anonymized.',
                                                        'order': 3,
```

(continues on next page)

(continued from previous page)

```
                                                    'values-regex': "([a-z0-9_\-\.
↪]{2,}@[a-z0-9_\-\.]{2,}\.[a-z]{2,},){0,}([a-z0-9_\-\.]{2,}@[a-z0-9_\-\.]{2,}\.[a-z]
↪{2,})",
                                                    'required':True
                                                },
                                                'abstract': {
                                                    'description': 'Abstract of␣
↪paper.',
                                                    'order': 4,
                                                    'value-regex': '[\\S\\s]{1,
↪5000}',
                                                    'required':True
                                                },
                                                'pdf': {
                                                    'description': 'Upload a PDF␣
↪file that ends with .pdf',
                                                    'order': 5,
                                                    'value-regex': 'upload',
                                                    'required':True
                                                }
                                            }
                                        }))
```

```
{'cdate': 1531339106644,
'ddate': None,
'duedate': 1562875092000,
'id': u'ICML.cc/2019/Conference/-/Submission',
'invitees': [u'everyone'],
'multiReply': None,
'noninvitees': [],
'nonreaders': [],
'process': None,
'rdate': None,
'readers': [u'everyone'],
'reply': {u'content': {u'abstract': {u'description': u'Abstract of paper.',
                                     u'order': 4,
                                     u'required': True,
                                     u'value-regex': u'[\\S\\s]{1,5000}'},
                       u'authorids': {u'description': u"Comma separated list of␣
↪author email addresses, lowercased, in the same order as above. For authors with␣
↪existing OpenReview accounts, please make sure that the provided email address(es)␣
↪match those listed in the author's profile. Please provide real emails; identities␣
↪will be anonymized.",
                                      u'order': 3,
                                      u'required': True,
                                      u'values-regex': u'([a-z0-9_\\-\\.]{2,}@[a-z0-
↪9_\\-\\.]{2,}\\.[a-z]{2,},){0,}([a-z0-9_\\-\\.]{2,}@[a-z0-9_\\-\\.]{2,}\\.[a-z]{2,})
↪'},
                       u'authors': {u'description': u'Comma separated list of author␣
↪names. Please provide real names; identities will be anonymized.',
                                    u'order': 2,
                                    u'required': True,
                                    u'values-regex': u'[^;,\\n]+(,[^,\\n]+)*'},
                       u'pdf': {u'description': u'Upload a PDF file that ends with .
↪pdf',
                                u'order': 5,
```

(continues on next page)

(continued from previous page)

```
                                u'required': True,
                                u'value-regex': u'upload'},
                          u'title': {u'description': u'Title of paper.',
                                     u'order': 1,
                                     u'required': True,
                                     u'value-regex': u'.{1,250}'}},
           u'forum': None,
           u'readers': {u'description': u'The users who will be allowed to read the␣
↪above content.',
                        u'values': [u'everyone']},
           u'replyto': None,
           u'signatures': {u'description': u'Your authorized identity to be␣
↪associated with the above content.',
                           u'values-regex': u'~.*'},
           u'writers': {u'values-regex': u'~.*'}},
'signatures': [u'ICML.cc/2019/Conference'],
'taskCompletionCount': None,
'transform': None,
'web': None,
'writers': [u'ICML.cc/2019/Conference']}
```

Once an invitation for submission with a future due date is created, this conference is listed on the Openreview home page under the header "Open for Submissions".

Openreview home page after invitation for submission with a future due date was created:

Conference home page:

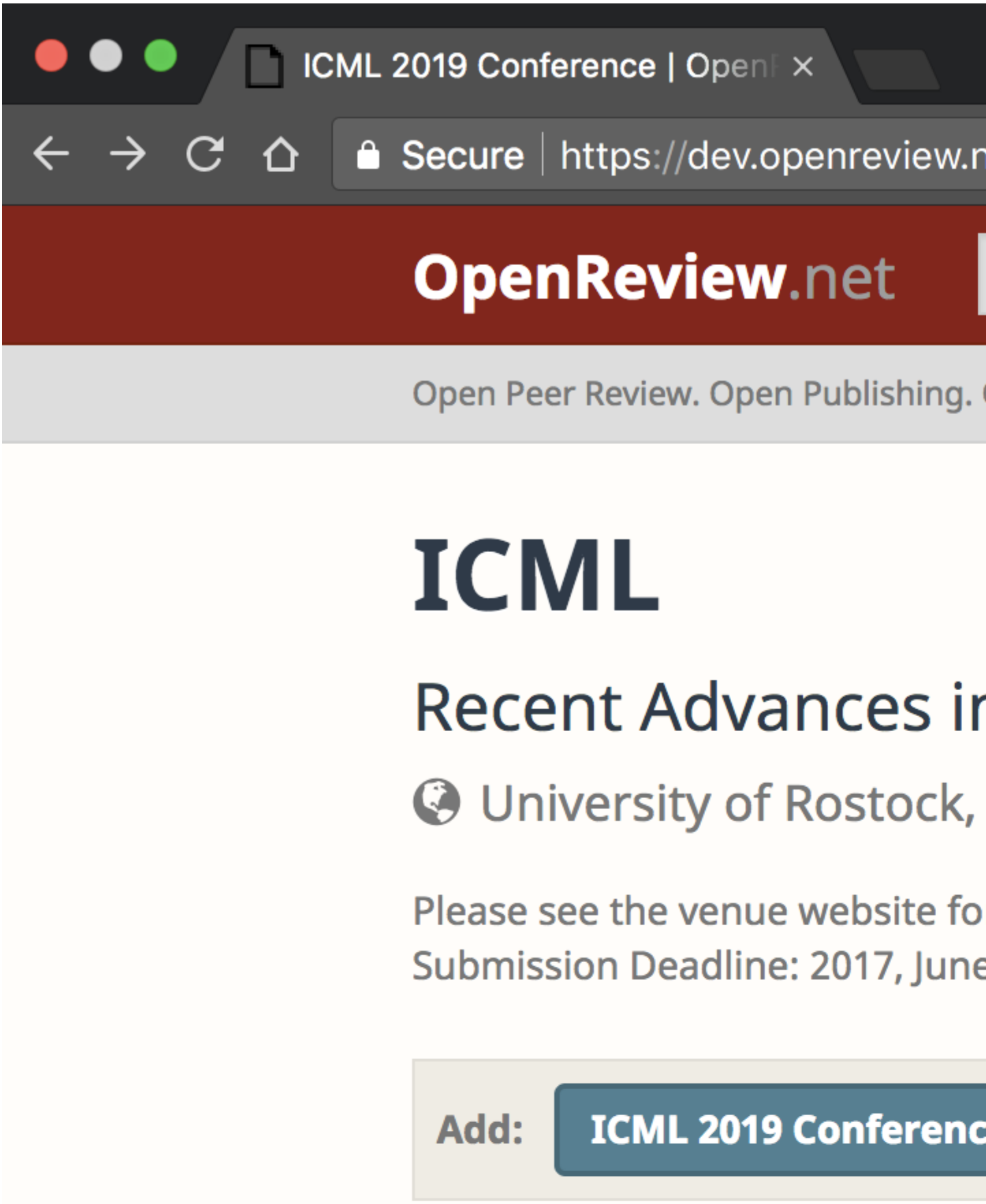
Create Invitation to Comment

Once an invitation for submission is created in your conference, you should be able to create an invitation to comment using the API. Creating an invitation to comment enables users to comment on a submission and reply to other's comments.:

```
>>> client.post_invitation(openreview.Invitation(id = 'ICML.cc/2019/Conference/-/
↪Comment',
                                                 readers = ['everyone'],
                                                 writers = ['ICML.cc/2019/Conference'],
                                                 signatures = ['ICML.cc/2019/Conference
↪'],
                                                 invitees = ['everyone'],
                                                  reply = {
                                                          'invitation': 'ICML.cc/2019/
↪Conference/-/Submission',
                                                          'content': {
                                                              'title': {
                                                                  'description':
↪'Comment title',
                                                                  'order': 1,
                                                                  'value-regex': '.*'
                                                              },
                                                              'comment': {
                                                                  'description':
↪'Comment',
                                                                  'order': 2,
                                                                  'value-regex': '.{0,
↪1000}'
```

(continues on next page)


Venues | OpenReview
Secure
https://dev.openreview.n
OpenReview.net
Open Peer Review. Open Publishing.
Open for Sub
ICLR 2018 Conf
ICML 2019 Con
All Venues
ICLR 2018


ICML 2019 Conference | Open
Secure
https://dev.openreview.n
OpenReview.net
Open Peer Review. Open Publishing.
ICML
Recent Advances in
University of Rostock,
Please see the venue website fo
Submission Deadline: 2017, June
Add:
ICML 2019 Conferenc
Submitted Papers

(continued from previous page)

```
                                                    }
                                                },
                                                'readers': {
                                                    'values': ['everyone']
                                                },
                                                'signatures': {
                                                    'values-regex':
↪'\\(anonymous\\)|~.*'
                                                },
                                                'writers': {
                                                    'values-regex':
↪'\\(anonymous\\)|~.*'
                                                }
                                            }))
```

```
{'cdate': 1531340152826,
'ddate': None,
'duedate': None,
'id': u'ICML.cc/2019/Conference/-/Comment',
'invitees': [u'everyone'],
'multiReply': None,
'noninvitees': [],
'nonreaders': [],
'process': None,
'rdate': None,
'readers': [u'everyone'],
'reply': {u'content': {u'comment': {u'description': u'Comment',
                                    u'order': 2,
                                    u'value-regex': u'.{0,1000}'},
                       u'title': {u'description': u'Comment title',
                                  u'order': 1,
                                  u'value-regex': u'.*'}},
          u'invitation': u'ICML.cc/2019/Conference/-/Submission',
          u'readers': {u'values': [u'everyone']},
          u'signatures': {u'values-regex': u'\\(anonymous\\)|~.*'},
          u'writers': {u'values-regex': u'\\(anonymous\\)|~.*'}},
'signatures': [u'ICML.cc/2019/Conference'],
'taskCompletionCount': None,
'transform': None,
'web': None,
'writers': [u'ICML.cc/2019/Conference']}
```

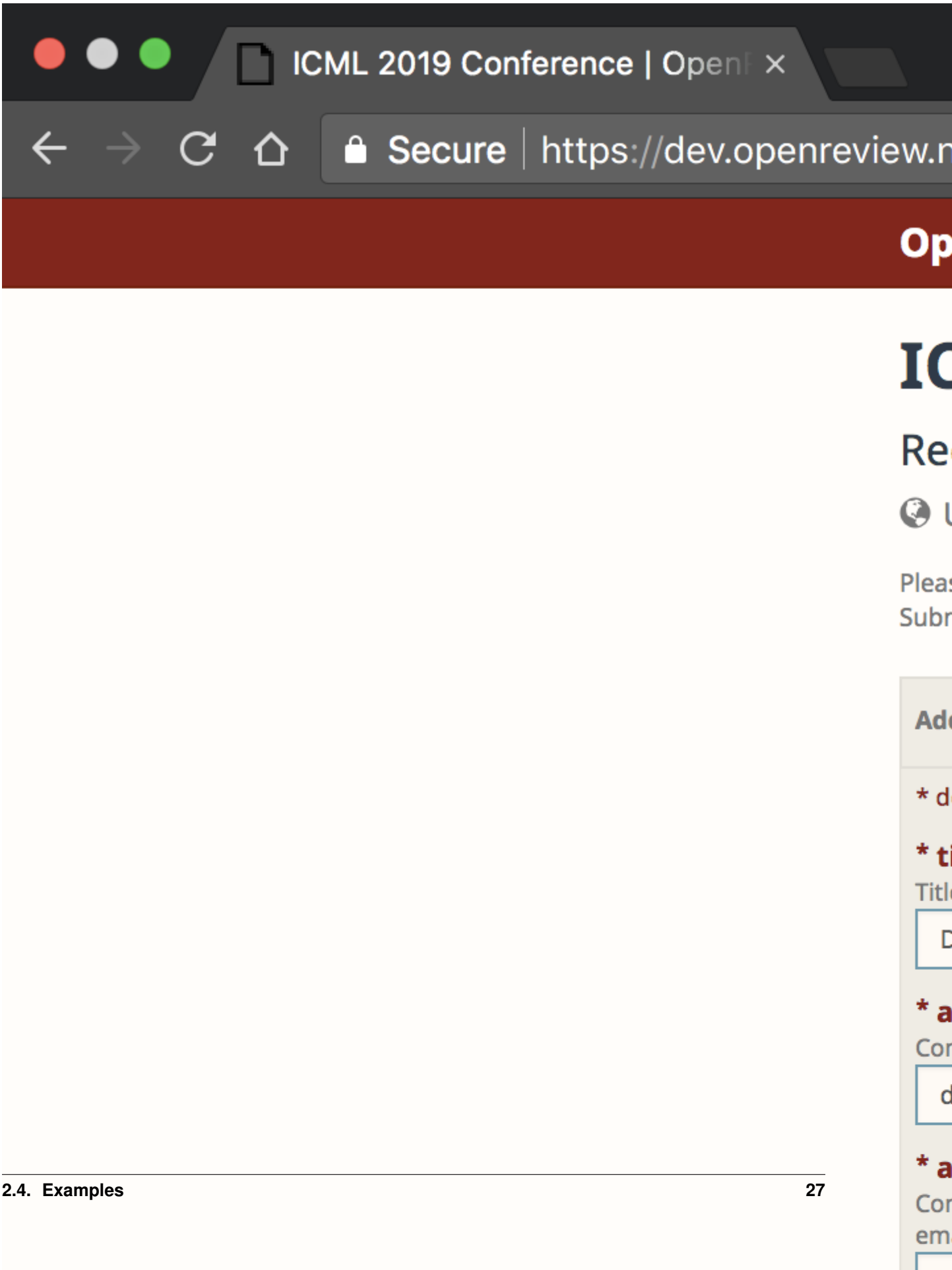
Making a Submission

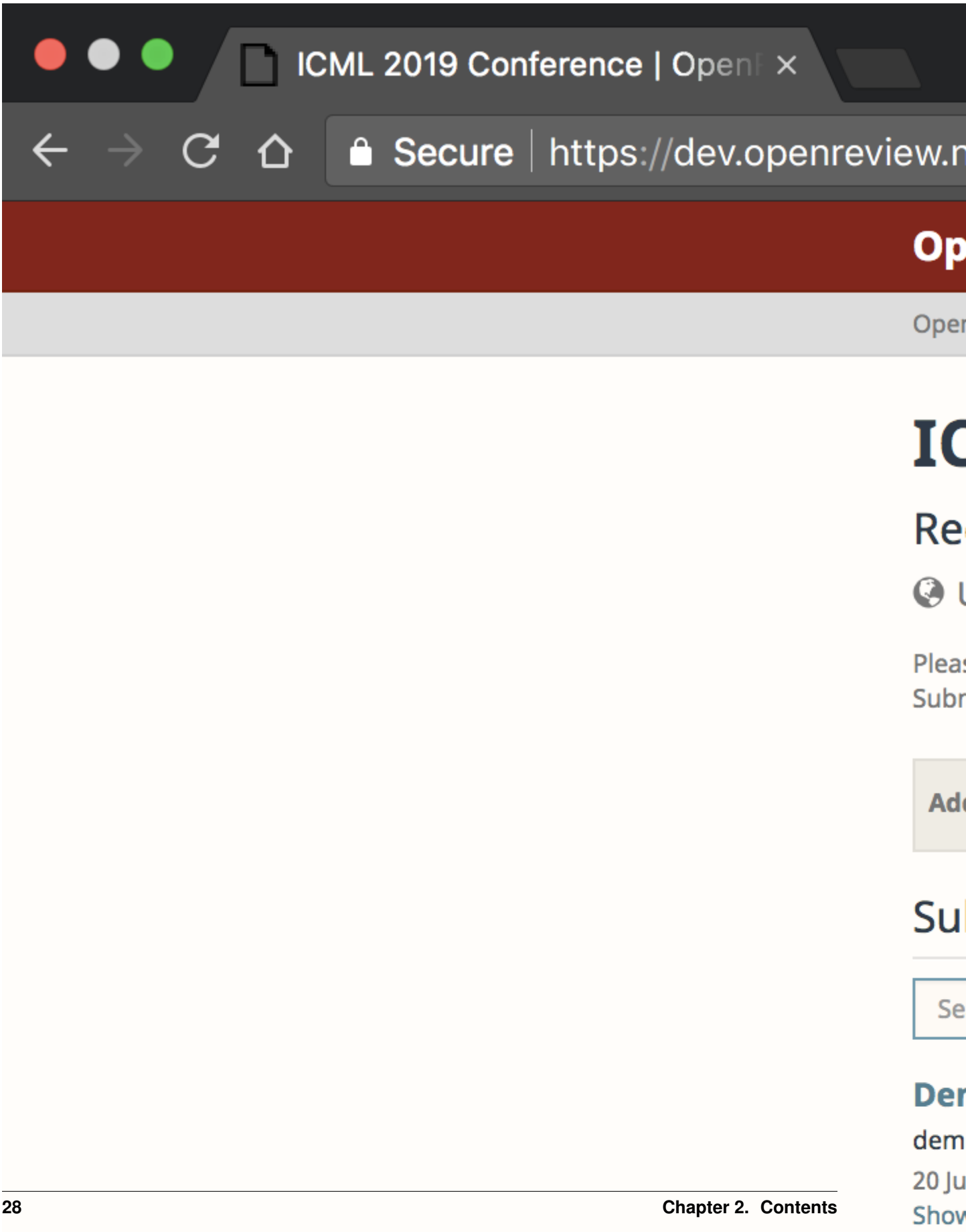
Once a submission invitation with a future due date is created, users with appropriate access can make a submission (e.g. submission of a research paper) using the Conference homepage.

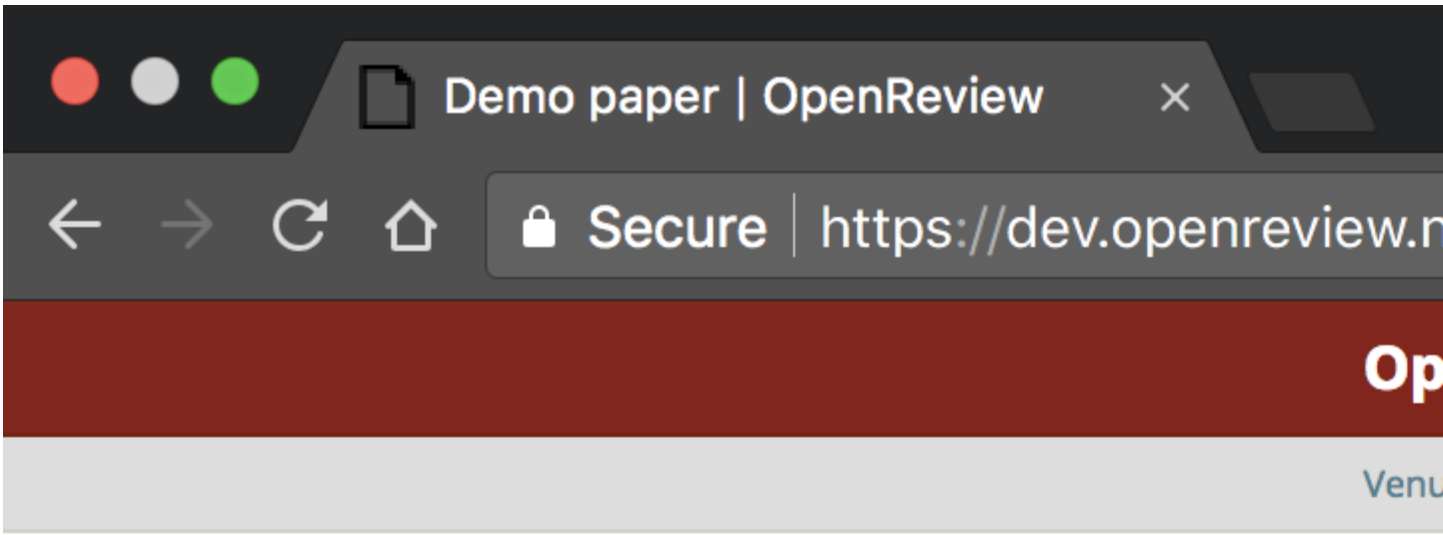
And, once a submission is made, a forum is created for that. This is what a forum looks like:

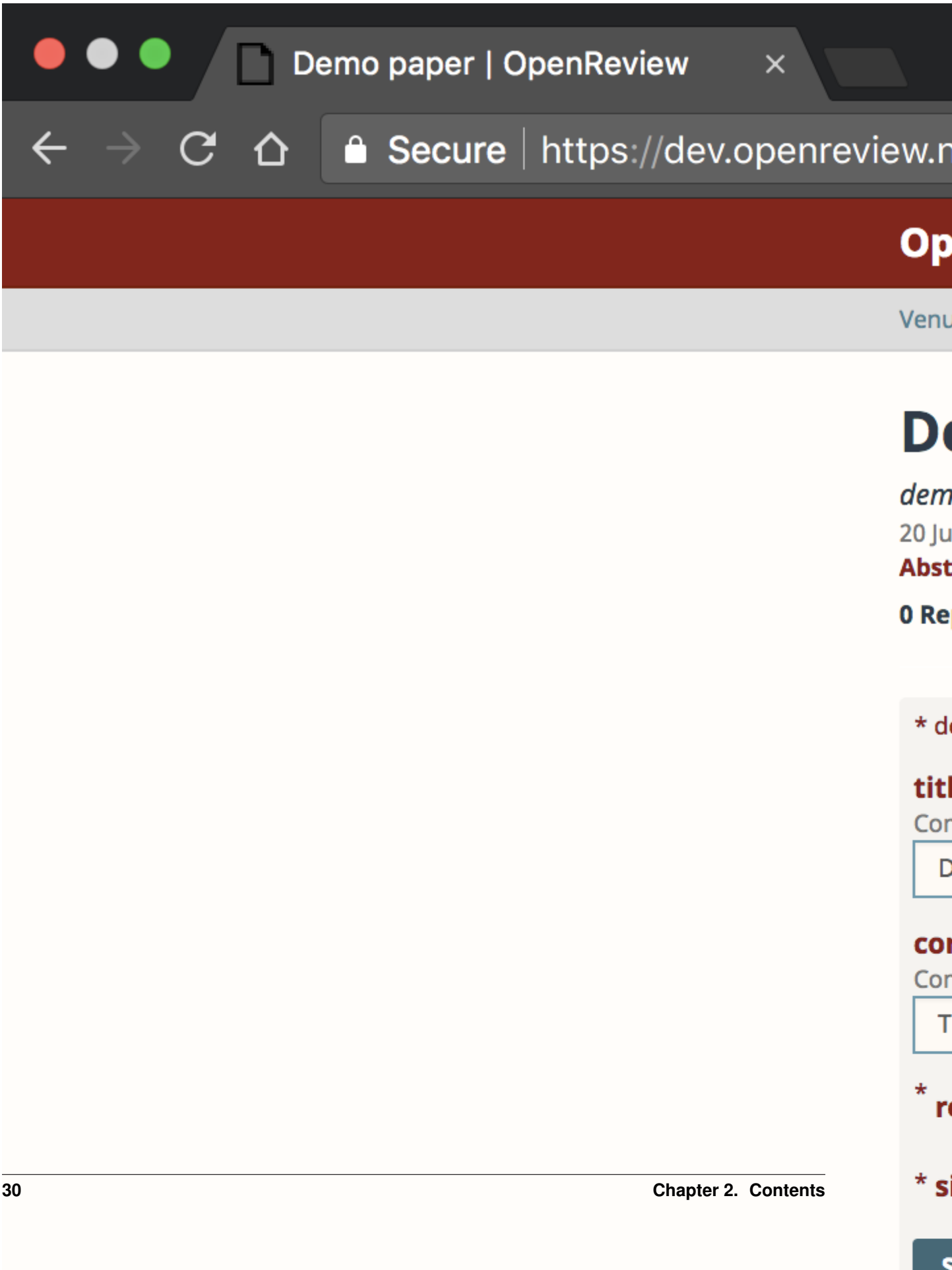
Commenting on a Submission

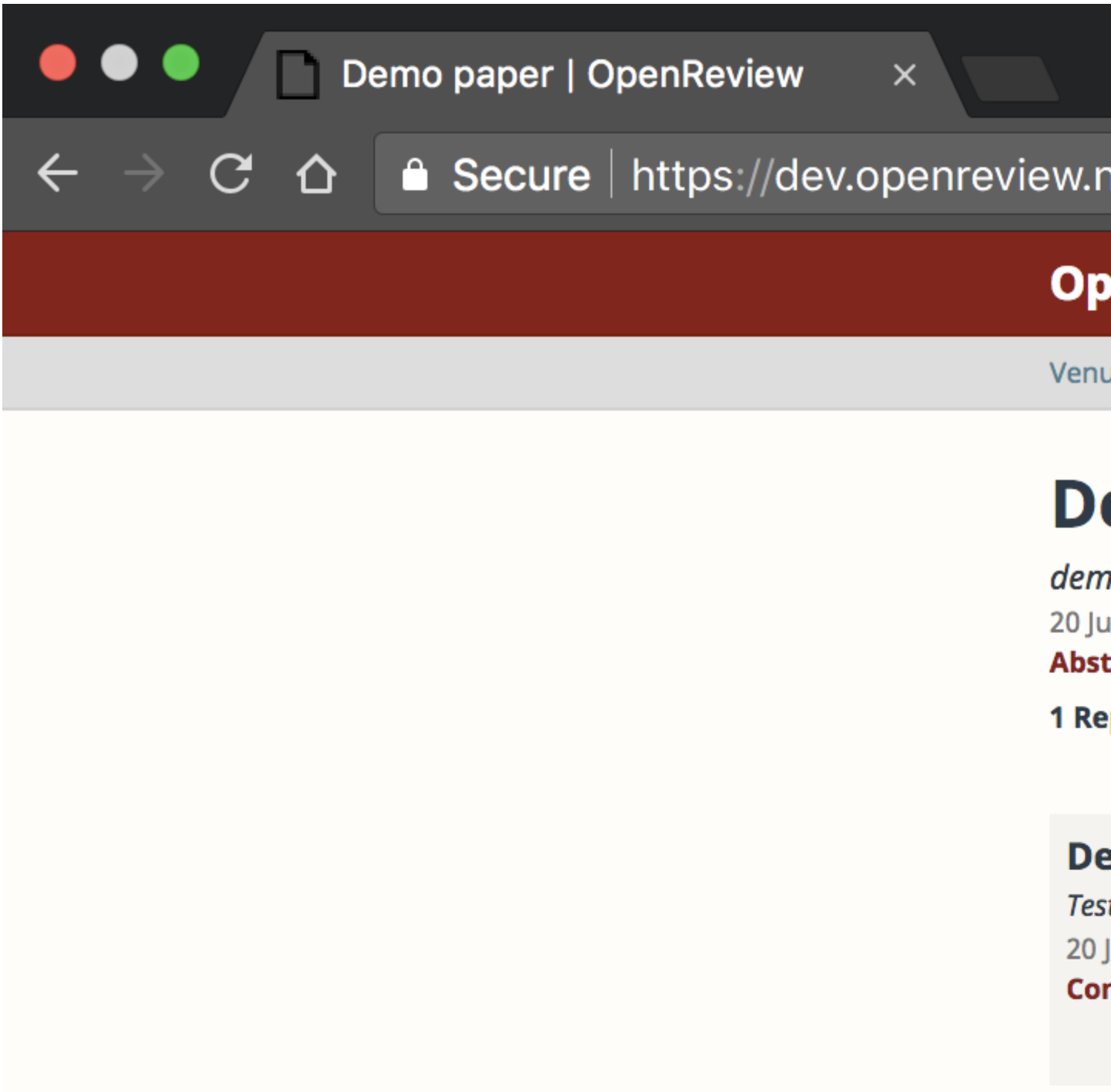
Users with appropriate access can comment on a submission and reply to other's comments (depending on the configuration settings of the conference).


ICML 2019 Conference | Open
Secure
https://dev.openreview.n


ICML 2019 Conference | Open
Secure
https://dev.openreview.n


Demo paper | OpenReview
Secure
https://dev.openreview.n


Demo paper | OpenReview
Secure
https://dev.openreview.n
Op
Venu
D
dem
20 Ju
Abst
0 Re
* d
titl
Cor
D
co
Cor
T
*
r
*
si


Demo paper | OpenReview
Secure
https://dev.openreview.n
Op
Venu
D
dem
20 Ju
Abst
1 Re
De
Test
20 J
Cor

Retrieving all notes given an invitation

Users can access all notes by using an invitation id. Important note: only the notes readable by the account the user logged in with can be accessed. Consider the following example which gets all the papers submitted to ICLR 2018:

```
>>>iclr18_submissions = client.get_notes(invitation="ICLR.cc/2018/Conference/-/
↪Submission")
```

Please note, get_notes() will return only the first 1000 notes. If you expect the number of notes returned to be more, you will require tools.iterget_notes() which returns an iterator and lets you access all the notes.

Retrieving all Public Comments for a conference

Comments, just like submissions, are saved as notes. So they are also accessible using get_notes(). Note that the invitation argument in get_notes() accepts regex. Consider the following example to get all the "Public Comments" made on submissions in ICLR 2019 conference.:

```
>>>iclr19_public_comments = client.get_notes(invitation="ICLR.cc/2019/Conference/-/
↪Paper.*/Public_Comment")
```

This code returns public comments made during the conference "ICLR.cc/2019/Conference" with invitation such as "ICLR.cc/2019/Conference/-/Paper1234/Public_Comment".

Please note that the invitation regex used in above example represents the convention that OpenReview follows while creating invitations for ICLR Conferences. So, the invitations do not necessarily have to follow this regex for all conferences or workshops.

Retrieving all Official Reviews for a conference

Like comments and submissions, reviews are also usually represented as Notes. Conferences often distinguish reviews written by official conference reviewers with the invitation suffix "Official_Review".

For example, the reviews in ICLR 2019 all have invitations with the following pattern:

```
ICLR.cc/2019/Conference/-/Paper.*/Official_Review
```

To retrieve the Official Reviews for a given ICLR 2019 paper, do the following:

```
>>> paper123_reviews = client.get_notes(invitation='ICLR.cc/2019/Conference/-/
↪Paper123/Official_Review')
```

The specific structure of the review's `content` field is determined by the conference, but a typical review's content will include fields like `title`, `review`, `rating`, and `confidence`:

```
>>> review0 = paper123_reviews[0]
>>> print(review0.content['rating'])
'8: Top 50% of accepted papers, clear accept'
```

Conferences as large as ICLR 2019 will often have a number of reviews that exceeds the default API limit. To retrieve all Official Reviews for all ICLR 2019 papers, create an iterator over reviews by doing the following:

```
>>> review_iterator = openreview.tools.iterget_notes(client, invitation='ICLR.cc/2019/
↪Conference/-/Paper.*/Official_Review')
>>> for review in review_iterator:
>>>     #do something
```

Retrieving comments made on a forum

All comments made on a particular forum/submission can be extracted like this:

```
>>>iclr19_forum_comments = client.get_notes(forum="<forum-id>")
```

Also, the public comments on a particular forum can be extracted like this:

```
>>>iclr19_forum_public_comments = client.get_notes(forum="<forum-id>", invitation=
↪"ICLR.cc/2019/Conference/-/Paper.*/Public_Comment")
```

Accessing data in comments

The data in a comment, or basically Notes objects, can be accessed like this:

```
>>>print(iclr19_forum_public_comments[0].content["title"])
>>>print(iclr19_forum_public_comments[0].content["comment"])
```

2.4.2 Getting all Venues

```
>>> import openreview
>>> from __future__ import print_function
>>> c = openreview.Client(baseurl='https://openreview.net')
>>> venues = openreview.get_all_venues(c)
>>> print(*venues,sep="\n")
    ICLR.cc
    auai.org/UAI
    NIPS.cc
    ICML.cc
    AKBC.ws
    BNMW_Workshop
    IEEE.org
    informatik.uni-rostock.de/Informatik_Rostock/Ubiquitous_Computing
    ISMIR.net
    swsa.semanticweb.org/ISWC
    machineintelligence.cc/MIC
    MIDL.amsterdam
    OpenReview.net/Anonymous_Preprint
    roboticsfoundation.org/RSS
```

2.4.3 Getting Submissions

All the Invitation Ids for Submissions can be retrieved like this:

```
>>> from __future__ import print_function
>>> import openreview
>>> c = openreview.Client(baseurl='https://openreview.net')
>>> invi = openreview.get_submission_invitations(c)
>>> print(*inv,sep="\n")
machineintelligence.cc/MIC/2018/Conference/-/Submission
machineintelligence.cc/MIC/2018/Abstract/-/Submission
ICLR.cc/2018/Workshop/-/Withdraw_Submission
```

(continues on next page)

(continued from previous page)

```
ICLR.cc/2018/Workshop/-/Withdrawn_Submission
aclweb.org/NAACL/2018/Preprint/-/Blind_Submission
aclweb.org/NAACL/2018/Preprint/-/Submission
ICLR.cc/2018/Conference/-/Withdrawn_Submission
ICLR.cc/2013/conference/-/submission
ICLR.cc/2014/-/submission/workshop
ICLR.cc/2014/-/submission/conference
ICLR.cc/2014/-/submission
ICML.cc/2013/PeerReview/-/submission
ICML.cc/2013/Inferning/-/submission
ICLR.cc/2013/-/submission
AKBC.ws/2013/-/submission
ICLR.cc/2016/workshop/-/submission
ICML.cc/2018/Workshop/NAMPI/-/Submission
NIPS.cc/2017/Workshop/Autodiff/-/Submission
ICLR.cc/2018/Conference/-/Blind_Submission
roboticsfoundation.org/RSS/2017/RCW_Workshop/-_Proceedings/-/Submission
ICML.cc/2018/Workshop/NAMPI/-/Blind_Submission
ISMIR.net/2018/WoRMS/-/Submission
ICLR.cc/2018/Conference/-/Submission
auai.org/UAI/2018/-/Blind_Submission
swsa.semanticweb.org/ISWC/2017/DeSemWeb/-/Submission
auai.org/UAI/2017/-/submission
auai.org/UAI/2018/-/Submission
auai.org/UAI/2017/-/blind-submission
NIPS.cc/2016/workshop/NAMPI/-/submission
NIPS.cc/2017/Workshop/MLITS/-/Submission
swsa.semanticweb.org/ISWC/2018/DeSemWeb/-/Submission
IEEE.org/2018/ITSC/-/Blind_Submission
MIDL.amsterdam/2018/Abstract/-/Submission
ICML.cc/2017/RML/-/Submission
ICLR.cc/2018/Workshop/-/Submission
OpenReview.net/Anonymous_Preprint/-/Submission
ICLR.cc/2017/workshop/-/submission
ECCV2016.org/BNMW/-/submission
IEEE.org/2018/ITSC/-/Submission
OpenReview.net/Anonymous_Preprint/-/Blind_Submission
ICML.cc/2018/RML/-/Submission
roboticsfoundation.org/RSS/2017/RCW_Workshop/-_Poster/-/Submission
NIPS.cc/2016/workshop/MLITS/-/submission
ICLR.cc/2014/conference/-/submission
ICLR.cc/2017/conference/-/submission
ICML.cc/2017/MLAV/-/Submission
ICML.cc/2017/WHI/-/Submission
ICLR.cc/2014/workshop/-/submission
cv-foundation.org/CVPR/2017/BNMW/-/Submission
MIDL.amsterdam/2018/Conference/-/Submission
ICML.cc/2018/ECA/-/Submission
```

Using these Invitation Ids we can extract the relevant notes like this:

```
>>> notes = c.get_notes(invitation='ICLR.cc/2018/Conference/-/Blind_Submission')
>>> print(notes[0])
{'cdate': 1518730187619,
 'content': {u'TL;DR': u'Neural phrase-based machine translation with linear decoding␣
↪time',
             u'_bibtex': u'@inproceedings{\nhuang2018towards,\ntitle={Towards Neural␣
↪Phrase-based Machine Translation},\nauthor={Po-Sen Huang and Chong Wang and Sitao␣
↪Huang and Dengyong Zhou and Li Deng},\nbooktitle={International Conference on␣
↪Learning Representations},\nyear={2018},\nurl={https://openreview.net/forum?
↪id=HktJec1RZ},\n}',
```

(continues on next page)

(continued from previous page)

```
            u'abstract': u'In this paper, we present Neural Phrase-based Machine␣
↪Translation (NPMT). Our method explicitly models the phrase structures in output␣
↪sequences using Sleep-WAke Networks (SWAN), a recently proposed segmentation-based␣
↪sequence modeling method. To mitigate the monotonic alignment requirement of SWAN,␣
↪we introduce a new layer to perform (soft) local reordering of input sequences.␣
↪Different from existing neural machine translation (NMT) approaches, NPMT does not␣
↪use attention-based decoding mechanisms.  Instead, it directly outputs phrases in a␣
↪sequential order and can decode in linear time. Our experiments show that NPMT␣
↪achieves superior performances on IWSLT 2014 German-English/English-German and␣
↪IWSLT 2015 English-Vietnamese machine translation tasks compared with strong NMT␣
↪baselines. We also observe that our method produces meaningful phrases in output␣
↪languages.',
            u'authorids': [u'huang.person@gmail.com',
                           u'chongw@google.com',
                           u'shuang91@illinois.edu',
                           u'dennyzhou@gmail.com',
                           u'l.deng@ieee.org'],
            u'authors': [u'Po-Sen Huang',
                         u'Chong Wang',
                         u'Sitao Huang',
                         u'Dengyong Zhou',
                         u'Li Deng'],
            u'keywords': [u'Neural Machine Translation',
                          u'Sequence to Sequence',
                          u'Sequence Modeling'],
            u'paperhash': u'huang|towards_neural_phrasebased_machine_translation',
            u'pdf': u'/pdf/b907f63a962c897b8193743aec2e9ac67eb5f79a.pdf',
            u'title': u'Towards Neural Phrase-based Machine Translation'},
 'ddate': None,
 'forum': u'HktJec1RZ',
 'forumContent': None,
 'id': u'HktJec1RZ',
 'invitation': u'ICLR.cc/2018/Conference/-/Blind_Submission',
 'nonreaders': [],
 'number': 143,
 'original': None,
 'overwriting': None,
 'readers': [u'everyone'],
 'referent': None,
 'replyto': None,
 'signatures': [u'ICLR.cc/2018/Conference'],
 'tcdate': 1509036001039,
 'tmdate': 1531951157137,
 'writers': [u'ICLR.cc/2018/Conference']}
```

2.4.4 Add evidence to a profile

Add DBLP link:

```
>>> updated_profile = client.update_profile(openreview.Profile(referent = '~Melisa_
↪TestBok1',
>>>                     invitation = '~/-/invitation',
>>>                     signatures = ['~Melisa_TestBok1'],
>>>                     content = {
>>>                      'dblp': 'http://dblp.org/mbok'
```

(continues on next page)

(continued from previous page)

```
>>>                     }))
```

Get references:

```
>>> references = client.get_references(referent = '~Melisa_TestBok1')
```

Add a history record:

```
>>> updated_profile = client.update_profile(openreview.Profile(referent = '~Melisa_
↪TestBok1',
>>>                     invitation = '~/-/invitation',
>>>                     signatures = ['~Melisa_TestBok1'],
>>>                     content = {
>>>                      'history': {
>>>                         'position': 'Developer',
>>>                         'institution': { 'name': 'UBA', 'domain': 'uba.ar'},
>>>                         'start': 2000,
>>>                         'end': 2006
>>>                      }
>>>                     }))
```

Add email:

```
>>> updated_profile = client.update_profile(openreview.Profile(referent = '~Melisa_
↪TestBok1',
>>>                     invitation = '~/-/invitation',
>>>                     signatures = ['~Melisa_TestBok1'],
>>>                     content = {
>>>                      'emails': 'test@mail.com'
>>>                     }))
```

Remove email:

```
>>> updated_profile = client.update_profile(openreview.Profile(referent = '~Melisa_
↪TestBok1',
>>>                     invitation = '~/-/invitation',
>>>                     signatures = ['~Melisa_TestBok1'],
>>>                     content = {},
>>>                     metaContent = {
>>>                      'emails': { 'values': ['test@mail.com'], 'weights': [-1] }
>>>                     }))
```

2.5 Help

For any queries contact info@openreview.net

2.6 Conference Builder

Tool to configure a conference from scratch, this builder is still in progress.

2.6.1 Create a builder

Create an instance of the conference builder. This requires to have an OpenReview client.

```
>>> builder = openreview.conference.ConferenceBuilder(client)
```

2.6.2 Set conference properties

Set the properties that we need to start the conference

```
>>> builder.set_conference_id('AKBC.ws/2019/Conference')
>>> builder.set_conference_name('Automated Knowledge Base Construction')
>>> builder.set_homepage_header({
>>>     'title': 'AKBC 2019',
>>>     'subtitle': 'Automated Knowledge Base Construction',
>>>     'location': 'Amherst, Massachusetts, United States',
>>>     'date': 'May 20 - May 21, 2019',
>>>     'website': 'http://www.akbc.ws/2019/',
>>>     'instructions': '<p><strong>Important Information</strong>\
>>>         <ul>\
>>>         <li>Note to Authors, Reviewers and Area Chairs: Please update your␣
↪OpenReview profile to have all your recent emails.</li>\
>>>         <li>AKBC 2019 Conference submissions are now open.</li>\
>>>         <li>For more details refer to the <a href="http://www.akbc.ws/2019/cfp/">
↪AKBC 2019 - Call for Papers</a>.</li>\
>>>         </ul></p> \
>>>         <p><strong>Questions or Concerns</strong></p>\
>>>         <p><ul>\
>>>         <li>Please contact the AKBC 2019 Program Chairs at \
>>>         <a href="mailto:info@akbc.ws">info@akbc.ws</a> with any questions or␣
↪concerns about conference administration or policy.</li>\
>>>         <li>Please contact the OpenReview support team at \
>>>         <a href="mailto:info@openreview.net">info@openreview.net</a> with any␣
↪questions or concerns about the OpenReview platform.</li>\
>>>         </ul></p>',
>>>     'deadline': 'Submission Deadline: Midnight Pacific Time, Friday, October 5,␣
↪2019'
>>> })
>>> conference = builder.get_result()
```

this will setup the home page of the conference: https://opereview.net/group?id=AKBC.ws/2019/Conference

2.6.3 Set Program Chairs

Set the members of the program chair committee

```
>>> conference.set_program_chairs(['pc@mail.com', 'pc2@mail.com'])
```

2.6.4 Open Submissions

This call will setup an invitation to submit papers following the conference type, in this case is double blind, and with the submission deadline for Friday, October 5, 2019

```
>>> invitation = conference.open_submissions(due_date = datetime.datetime(2019, 10, 5,
↪ 23, 59))
```

2.6.5 Close Submissions

The submission will be closed when the current date passes the due date specified in open submissions call. In this case it will force the close of the submissions and it will disable the adition and edition of the papers by the authors. They can not edit the submission anymore

```
>>> invitation = conference.close_submissions()
```

2.6.6 Recruit Reviewers

Invite reviewers or area chairs to be part of the conference committee

```
>>> conference.recruit_reviewers(['mbok@mail.com', 'mohit@mail.com'])
```

This will send an email to the reviewers asking for a response to accept or reject the invitation. You can specify the text of the email:

```
>>> title = 'Invitation to review'
>>> message = '''
>>> Thank you for accepting our invitation to be an Area Chair for the AKBC. With␣
↪this email, we are inviting you to log on to the OpenReview.
>>>
>>> To ACCEPT the invitation, please click on the following link:
>>>
>>> {accept_url}
>>>
>>> If you changed your mind please DECLINE the invitation by clicking on the␣
↪following link:
>>>
>>> {decline_url}
>>>
>>> We'd appreciate an answer within five days. If you have any questions please␣
↪contact AKBC Program Chairs at <program-chairs@mail.com>.
>>>
>>> We are looking forward to your reply.
>>>
>>> AKBC Program Chairs
>>>
>>> '''
>>> conference.recruit_reviewers(emails = ['ac1@mail.com', 'ac2@mail.com'], title =␣
↪title, message = message, reviewers_name = 'Area_Chairs')
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

o

t

Index

R

S

T

U