Indico Documentation

Release 2.3-dev

Indico Team

Mar 11, 2020
Welcome to Indico’s documentation. This documentation is split into several parts, from installing Indico to developing Indico plugins. To dive into the internals of Indico, check out the API documentation. Read more about Indico in our official website.
Installation

To simply install and use Indico, follow the *production installation instructions*. For those who are interested in developing new features and plugins for Indico, check out the *development installation instructions*.

### 1.1 Installation guides

To simply install and use Indico, follow the *production installation instructions*. For those who are interested in developing new features and plugins for Indico, check out the *development installation instructions*.

#### 1.1.1 Production

We provide guides to install Indico on CentOS and Debian systems. While other distributions are not officially supported, they should work fine, but the installation steps (especially package names) may need some slight adjustments.

Our guides cover a single-machine installation where Indico, Celery, Redis and PostgreSQL run on the same machine. This should be fine for almost all Indico instances, but adapting the steps to multiple machines is not particularly hard either.

**CentOS7 / CC7**

Except for minor differences, these guides apply to both vanilla CentOS7 and the CERN flavor of CentOS, CC7 (CentOS CERN 7).

**nginx**

*Note:* Please note that you **must** use Apache if you intend to use SSO using Shibboleth/SAML/ADFS. If that’s not the case because you do not use SSO at all or use e.g. OAuth, we recommend using nginx.
1. Enable EPEL

```bash
yum install -y epel-release
```

**Note:** If you use CC7, EPEL is already enabled and this step is not necessary.

2. Install Packages

Edit `/etc/yum.repos.d/CentOS-Base.repo` and add `exclude=postgresql*` to the `[base]` and `[updates]` sections, as described in the PostgreSQL wiki.

```bash
yum install -y https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/
    →pgdg-centos96-9.6-3.noarch.rpm
yum install -y postgresql96 postgresql96-server postgresql96-libs postgresql96-devel
    →postgresql96-contrib
yum install -y gcc redis nginx uwsgi uwsgi-plugin-python2
yum install -y python-devel python-virtualenv libjpeg-turbo-devel libxslt-devel
    →libxml2-devel libffi-devel pcre-devel libyaml-devel
/usr/pgsql-9.6/bin/postgresql96-setup initdb
systemctl start postgresql-9.6.service redis.service
```

3. Create a Database

We create a user and database for indico and enable the necessary Postgres extensions (which can only be done by the Postgres superuser).

```bash
su - postgres -c 'createuser indico'
su - postgres -c 'createdb -O indico indico'
su - postgres -c 'psql indico -c "CREATE EXTENSION unaccent; CREATE EXTENSION pg_trgm;"'
```

**Warning:** Do not forget to setup a cronjob that creates regular database backups once you start using Indico in production!

4. Configure uWSGI & nginx

The default uWSGI and nginx configuration files should work fine in most cases.

```bash
cat > /etc/uwsgi.ini <<'EOF'
[uwsgi]
uid = indico
gid = nginx
umask = 027

processes = 4
enable-threads = true
chmod-socket = 770
socket = /opt/indico/web/uwsgi.sock
```

(continues on next page)
stats = /opt/indico/web/uwsgi-stats.sock
protocol = uwsgi
master = true
auto-procname = true
procname-prefix-spaced = indico
disable-logging = true
plugin = python
single-interpreter = true
touch-reload = /opt/indico/web/indico.wsgi
wsgi-file = /opt/indico/web/indico.wsgi
virtualenv = /opt/indico/.venv
vacuum = true
buffer-size = 20480
memory-report = true
max-requests = 2500
harakiri = 900
harakiri-verbose = true
reload-on-rss = 2048
evil-reload-on-rss = 8192
EOF

**Note:** Replace `YOURHOSTNAME` in the next file with the hostname on which your Indico instance should be available, e.g. `indico.yourdomain.com`

```bash
cat > /etc/nginx/conf.d/indico.conf <<'EOF'
server {
    listen 80;
    listen [::]:80;
    server_name YOURHOSTNAME;
    return 301 https://$server_name$request_uri;
}

server {
    listen *:443 ssl http2;
    listen [::]:443 ssl http2 default ipv6only=on;
    server_name YOURHOSTNAME;

    ssl on;
    ssl_certificate /etc/ssl/indico/indico.crt;
    ssl_certificate_key /etc/ssl/indico/indico.key;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    include /etc/ssl/openssl.cnf;
    # See https://github.com/pacmpl/nginx-ssl-config-inject
}
EOF
```

1.1. Installation guides
ssl_prefer_server_ciphers on;

access_log /opt/indico/log/nginx/access.log combined;
error_log /opt/indico/log/nginx/error.log;

location /.xsf/indico/ {
    internal;
    alias /opt/indico/;
}

location ~ ^/(images|fonts)(.*)/(.+?)(__v[0-9a-f]+)?\.([^.]+)$ {
    alias /opt/indico/web/static/$1$2/$3.$5;
    access_log off;
}

location ~ ^/(css|dist|images|fonts)/(.*)$ {
    alias /opt/indico/web/static/$1/$2;
    access_log off;
}

location /robots.txt {
    alias /opt/indico/web/static/robots.txt;
    access_log off;
}

location / {
    root /var/empty/nginx;
    include /etc/nginx/uwsgi_params;
    uwsgi_pass unix:/opt/indico/web/uwsgi.sock;
    uwsgi_param UWSGI_SCHEME $scheme;
    uwsgi_read_timeout 15m;
    uwsgi_buffers 32 32k;
    uwsgi_busy_buffers_size 128k;
    uwsgi_hide_header X-Sendfile;
    client_max_body_size 1G;
}
}
EOF

5. Create an SSL Certificate

First, create the folders for the certificate/key and set restrictive permissions on them:

```
mkdir /etc/ssl/indico
chown root:root /etc/ssl/indico/
chmod 700 /etc/ssl/indico
```

If you are just trying out Indico you can simply use a self-signed certificate (your browser will show a warning which you will have to confirm when accessing your Indico instance for the first time).

**Note:** Do not forget to replace YOURHOSTNAME with the same value you used above
openssl req -x509 -nodes -newkey rsa:4096 -subj /CN=YOURHOSTNAME -keyout /etc/ssl/indico/indico.key -out /etc/ssl/indico/indico.crt

While a self-signed certificate works for testing, it is not suitable for a production system. You can either buy a certificate from any commercial certification authority or get a free one from Let’s Encrypt.

**Note:** There’s an optional step later in this guide to get a certificate from Let’s Encrypt. We can’t do it right now since the nginx config references a directory yet to be created, which prevents nginx from starting.

### 6. Configure SELinux

Indico works fine with SELinux enabled, but you need to load a custom SELinux module to tell SELinux about Indico’s files and how they should be handled.

```bash
cat > /tmp/indico.cil <<'EOF'
; define custom type that logrotate can access
(type indico_log_t)
(typeattributeset file_type (indico_log_t))
(typeattributeset logfile (indico_log_t))
(roletype object_r indico_log_t)

; allow logrotate to reload systemd services
(allow logrotate_t init_t (service (start)))
(allow logrotate_t policykit_t (dbus (send_msg)))
(allow policykit_t logrotate_t (dbus (send_msg)))

; make sure the uwsgi socket is writable by the webserver
(typetransition unconfined_service_t usr_t sock_file "uwsgi.sock" httpd_sys_rw_content_t)
(filecon "/opt/indico/web/uwsgi\.sock" socket (system_u object_r httpd_sys_rw_content_t))

; set proper types for our log dirs
(filecon "/opt/indico/log/(.*)?" any (system_u object_r indico_log_t))
(filecon "/opt/indico/log/nginx/(.*)?" any (system_u object_r httpd_log_t))
EOF

semodule -i /tmp/indico.cil
```

### 7. Install Indico

Celery runs as a background daemon. Add a systemd unit file for it:

```bash
cat > /etc/systemd/system/indico-celery.service <<'EOF'
[Unit]
Description=Indico Celery
After=network.target

[Service]
ExecStart=/opt/indico/.venv/bin/indico celery worker -B
Restart=always
SyslogIdentifier=indico-celery
User=indico
EOF
```

(continues on next page)
Indico Documentation, Release 2.3-dev

(continued from previous page)

Group=nginx
UMask=0027
Type=simple
KillMode=mixed
TimeoutStopSec=300
[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload

Now create a user that will be used to run Indico and switch to it:
useradd -rm -g nginx -d /opt/indico -s /bin/bash indico
su - indico

You are now ready to install Indico:
Note: If you need to migrate from Indico 1.2, you must install Indico 2.0, regardless of what the latest Indico version
is. If this is the case for you, replace the last command in the block below with pip install 'indico<2.1'
virtualenv ~/.venv
source ~/.venv/bin/activate
export PATH="$PATH:/usr/pgsql-9.6/bin"
pip install -U pip setuptools
pip install indico

8. Configure Indico
Once Indico is installed, you can run the configuration wizard. You can keep the defaults for most options, but make
sure to use https://YOURHOSTNAME when prompted for the Indico URL. Also specify valid email addresses when
asked and enter a valid SMTP server Indico can use to send emails. When asked for the default timezone make sure
this is the main time zone used in your Indico instance.
indico setup wizard

Now finish setting up the directory structure and permissions:
mkdir ~/log/nginx
chmod go-rwx ~/* ~/.[^.]*
chmod 710 ~/ ~/archive ~/cache ~/log ~/tmp
chmod 750 ~/web ~/.venv
chmod g+w ~/log/nginx
restorecon -R ~/
echo -e "\nSTATIC_FILE_METHOD = ('xaccelredirect', {'/opt/indico': '/.xsf/indico'})" >
˓→> ~/etc/indico.conf

9. Create database schema
Finally you can create the database schema and switch back to root:

8

Chapter 1. Installation


10. Launch Indico

You can now start Indico and set it up to start automatically when the server is rebooted:

```bash
systemctl restart uwsgi.service nginx.service indico-celery.service
systemctl enable uwsgi.service nginx.service postgresql-9.6.service redis.service
indico-celery.service
```

11. Open the Firewall

```bash
firewall-cmd --permanent --add-port 443/tcp --add-port 80/tcp
firewall-cmd --reload
```

**Note:** This is only needed if you use CC7 as CentOS7 has no firewall enabled by default

12. Optional: Get a Certificate from Let’s Encrypt

To avoid ugly SSL warnings in your browsers, the easiest option is to get a free certificate from Let’s Encrypt. We also enable the cronjob to renew it automatically:

```bash
yum install -y python-certbot-nginx
certbot --nginx --rsa-key-size 4096 --no-redirect --staple-ocsp -d YOURHOSTNAME
rm -rf /etc/ssl/indico
systemctl start certbot-renew.timer
systemctl enable certbot-renew.timer
```

13. Create an Indico user

Access `https://YOURHOSTNAME` in your browser and follow the steps displayed there to create your initial user.

14. Install TeXLive

Follow the *LaTeX install guide* to install TeXLive so Indico can generate PDF files in various places.

**Apache**

1. Enable EPEL

```bash
yum install -y epel-release
```
2. Install Packages

Edit `/etc/yum.repos.d/CentOS-Base.repo` and add `exclude=postgresql*` to the `[base]` and `[updates]` sections, as described in the PostgreSQL wiki.

```
yum install -y https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/
   → pgdg-centos96-9.6-3.noarch.rpm
yum install -y postgresql96 postgresql96-server postgresql96-libs postgresql96-devel
   → postgresql96-contrib
yum install -y httpd mod_proxy_uwsgi mod_ssl mod_xsendfile
yum install -y gcc redis uwsgi uwsgi-plugin-python2
yum install -y python-devel python-virtualenv libjpeg-turbo-devel libxts1t-devel
   → libxml2-devel libffi-devel pcres-devel libyaml-devel
/usr/pgsql-9.6/bin/postgresql96-setup initdb
systemctl start postgresql-9.6.service redis.service
```

3. Create a Database

We create a user and database for indico and enable the necessary Postgres extensions (which can only be done by the Postgres superuser)

```
su - postgres -c 'createuser indico'
su - postgres -c 'createdb -O indico indico'
su - postgres -c 'psql indico -c "CREATE EXTENSION unaccent; CREATE EXTENSION pg_trgm;"'
```

**Warning:** Do not forget to setup a cronjob that creates regular database backups once you start using Indico in production!

4. Configure uWSGI & Apache

The default uWSGI and Apache configuration files should work fine in most cases.

```
cat > /etc/uwsgi.ini <<'EOF'
[uwsgi]
uid = indico
gid = apache
umask = 027

processes = 4
enable-threads = true
socket = 127.0.0.1:8008
stats = /opt/indico/web/uwsgi-stats.sock
protocol = uwsgi

master = true
auto-procname = true
EOF
```

(continues on next page)
procname-prefix-spaced = indico
disable-logging = true
plugin = python
single-interpreter = true
touch-reload = /opt/indico/web/indico.wsgi
wsgi-file = /opt/indico/web/indico.wsgi
virtualenv = /opt/indico/.venv
vacuum = true
buffer-size = 20480
memory-report = true
max-requests = 2500
harakiri = 900
harakiri-verbose = true
reload-on-rss = 2048
evil-reload-on-rss = 8192
EOF

Note: Replace YOURHOSTNAME in the next files with the hostname on which your Indico instance should be available, e.g. indico.yourdomain.com

cat > /etc/httpd/conf.d/indico-sslredir.conf <<'EOF'
<VirtualHost *:80>
  ServerName YOURHOSTNAME
  RewriteEngine On
  RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>
EOF

cat > /etc/httpd/conf.d/indico.conf <<'EOF'
<VirtualHost *:443>
  ServerName YOURHOSTNAME
  DocumentRoot "/var/empty/apache"
  SSLEngine on
  SSLCertificateFile /etc/ssl/indico/indico.crt
  SSLCertificateChainFile /etc/ssl/indico/indico.crt
  SSLCertificateKeyFile /etc/ssl/indico/indico.key
  SSLProtocol all -SSLv2 -SSLv3
  SSLHonorCipherOrder on
  XSendFile on
  XSendFilePath /opt/indico
</VirtualHost>
EOF

1.1. Installation guides
Now enable the uwsgi proxy module in apache:

```
echo 'LoadModule proxy_uwsgi_module modules/mod_proxy_uwsgi.so' > /etc/httpd/conf.modules.d/proxy_uwsgi.conf
```

### 5. Create an SSL Certificate

First, create the folders for the certificate/key and set restrictive permissions on them:

```
mkdir /etc/ssl/indico
chown root:root /etc/ssl/indico/
chmod 700 /etc/ssl/indico
```

If you are just trying out Indico you can simply use a self-signed certificate (your browser will show a warning which you will have to confirm when accessing your Indico instance for the first time).

**Note:** Do not forget to replace `YOURHOSTNAME` with the same value you used above

```
openssl req -x509 -nodes -newkey rsa:4096 -subj /CN=YOURHOSTNAME -keyout /etc/ssl/indico/indico.key -out /etc/ssl/indico/indico.crt
```

While a self-signed certificate works for testing, it is not suitable for a production system. You can either buy a certificate from any commercial certification authority or get a free one from Let’s Encrypt.

**Note:** There’s an optional step later in this guide to get a certificate from Let’s Encrypt. We can’t do it right now since the Apache config references a directory yet to be created, which prevents Apache from starting.
6. Configure SELinux

Indico works fine with SELinux enabled, but you need to load a custom SELinux module to tell SELinux about Indico’s files and how they should be handled.

```bash
cat > /tmp/indico.cil <<'EOF'
; define custom type that logrotate can access
(type indico_log_t)
(typeattributeset file_type (indico_log_t))
(typeattributeset logfile (indico_log_t))
(roletype object_r indico_log_t)

; allow logrotate to reload systemd services
(allow logrotate_t init_t (service (start)))
(allow logrotate_t policykit_t (dbus (send_msg)))
(allow policykit_t logrotate_t (dbus (send_msg)))

; make sure the uwsgi socket is writable by the webserver
(typetransition unconfined_service_t usr_t sock_file "uwsgi.sock" httpd_sys_rw_content_t)
(filecon "/opt/indico/web/uwsgi\.sock" socket (system_u object_r httpd_sys_rw_content_t ((s0)(s0))))

; set proper types for our log dirs
(filecon "/opt/indico/log(/.*)?" any (system_u object_r indico_log_t ((s0)(s0))))
(filecon "/opt/indico/log/apache(/.*)?" any (system_u object_r httpd_log_t ((s0)(s0))))

EOF
semodule -i /tmp/indico.cil
```

7. Install Indico

Celery runs as a background daemon. Add a systemd unit file for it:

```bash
cat > /etc/systemd/system/indico-celery.service <<'EOF'
[Unit]
Description=Indico Celery
After=network.target

[Service]
ExecStart=/opt/indico/.venv/bin/indico celery worker -B
Restart=always
SyslogIdentifier=indico-celery
User=indico
Group=apache
UMask=0027
Type=simple
KillMode=mixed
TimeoutStopSec=300

[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload
```

Now create a user that will be used to run Indico and switch to it:
You are now ready to install Indico:

Note: If you need to migrate from Indico 1.2, you must install Indico 2.0, regardless of what the latest Indico version is. If this is the case for you, replace the last command in the block below with `pip install 'indico<2.1'`.

```
# Create virtual environment
virtualenv ~/.venv
source ~/.venv/bin/activate
export PATH=$PATH:/usr/pgsql-9.6/bin
pip install -U pip setuptools
pip install indico
```

### 8. Configure Indico

Once Indico is installed, you can run the configuration wizard. You can keep the defaults for most options, but make sure to use `https://YOURHOSTNAME` when prompted for the Indico URL. Also specify valid email addresses when asked and enter a valid SMTP server Indico can use to send emails. When asked for the default timezone make sure this is the main timezone used in your Indico instance.

```
indico setup wizard
```

Now finish setting up the directory structure and permissions:

```
# Create directories
mkdir ~/log/apache
chmod go-rwx ~/[^.]*
chmod 710 ~/archive ~/cache ~/log ~/tmp
chmod 750 ~/web ~/.venv
chmod g+w ~/log/apache
restorecon -R ~/
# Add setting to Indico configuration
echo -e "\nSTATIC_FILE_METHOD = 'xsendfile'" >> ~/etc/indico.conf
```

### 9. Create database schema

Finally you can create the database schema and switch back to `root`:

```
indico db prepare
exit
```

### 10. Launch Indico

You can now start Indico and set it up to start automatically when the server is rebooted:

```
systemctl restart uwsgi.service httpd.service indico-celery.service
systemctl enable uwsgi.service httpd.service postgresql-9.6.service redis.service
```

```
˓→indico-celery.service
```
11. Open the Firewall

```
firewall-cmd --permanent --add-port 443/tcp --add-port 80/tcp
firewall-cmd --reload
```

**Note:** This is only needed if you use CC7 as CentOS7 has no firewall enabled by default.

12. Optional: Get a Certificate from Let’s Encrypt

To avoid ugly SSL warnings in your browsers, the easiest option is to get a free certificate from Let’s Encrypt. We also enable the cronjob to renew it automatically:

```
yum install -y python-certbot-apache
certbot --apache --rsa-key-size 4096 --no-redirect --staple-ocsp -d YOURHOSTNAME
rm -rf /etc/ssl/indico
systemctl start certbot-renew.timer
systemctl enable certbot-renew.timer
```

13. Create an Indico user

Access `https://YOURHOSTNAME` in your browser and follow the steps displayed there to create your initial user.

14. Install TeXLive

Follow the *LaTeX install guide* to install TeXLive so Indico can generate PDF files in various places.

**Optional: Shibboleth**

If your organization uses Shibboleth/SAML-based SSO, follow these steps to use it in Indico:

1. Install Shibboleth

Add the Shibboleth yum repository:

```
curl -fsSL -o /etc/yum.repos.d/shibboleth.repo 'https://shibboleth.net/cgi-bin/sp_˓
   →repo.cgi?platform=CentOS_7'
```

Now install Shibboleth itself. When prompted to accept the GPG key of the Shibboleth yum repo, confirm the prompt.

```
setsebool httpd_can_network_connect 1
yum install -y shibboleth xmltooling-schemas opensaml-schemas
```
2. Configure Shibboleth

This is outside the scope of this documentation and depends on your environment (Shibboleth, SAML, ADFS, etc). Please contact whoever runs your SSO infrastructure if you need assistance.

3. Enable Shibboleth in Apache

Add the following code to your `/etc/httpd/conf.d/indico.conf` right before the `AliasMatch` lines:

```
<LocationMatch "^(/Shibboleth\.sso|/login/shib-sso/shibboleth)"
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  ShibExportAssertion Off
  Require valid-user
</LocationMatch>
```

4. Enable Shibboleth in Indico

Add the following code to your `/opt/indico/etc/indico.conf`:

```
# SSO
AUTH_PROVIDERS = {
  'shib-sso': {
    'type': 'shibboleth',
    'title': 'SSO',
    'attrs_prefix': 'ADFS_',
    'callback_uri': '/login/shib-sso/shibboleth',
    # 'logout_uri': 'https://login.yourcompany.tld/logout'
  }
}
IDENTITY_PROVIDERS = {
  'shib-sso': {
    'type': 'shibboleth',
    'title': 'SSO',
    'identifier_field': 'ADFS_LOGIN',
    'mapping': {
      'affiliation': 'ADFS_HOMEINSTITUTE',
      'first_name': 'ADFS_FIRSTNAME',
      'last_name': 'ADFS_LASTNAME',
      'email': 'ADFS_EMAIL',
      'phone': 'ADFS_PHONENUMBER'
    },
    'trusted_email': True
  }
}
```

The values for `attrs_prefix`, `mapping` and `identifier_field` may be different in your environment. Uncomment and set `logout_uri` if your SSO infrastructure provides a logout URL (usually used to log you out from all applications).

If you only want to use SSO, without allowing people to login locally using username/password, disable it by setting `LOCAL_IDENTITIES = False` in `indico.conf`. 

---

**Chapter 1. Installation**
Warning: We assume that emails received from SSO are already validated. If this is not the case, make sure to disable `trusted_email` which will require email validation in Indico when logging in for the first time. Otherwise people could take over the account of someone else by using their email address!

Note: The example config is rather simple and only accesses data from SSO during login. This is not sufficient for advanced features such as automatic synchronization of names, affiliations and phone numbers or using centrally managed groups. To use these features, you need to use e.g. the LDAP identity provider and use the information received via SSO to retrieve the user details from LDAP. If you need assistance with this, feel free to ask us on IRC (#indico @ Freenode) or via e-mail (indico-team@cern.ch).

Note: Please note that you **must** use Apache if you intend to use SSO using Shibboleth/SAML/ADFS. If that’s not the case because you do not use SSO at all or use e.g. OAuth, we recommend using nginx.

**Debian / Ubuntu**

Except for minor differences, this guide applies to both Debian and Ubuntu. It has been tested with Debian 8 (Jessie), Debian 9 (Stretch) and Ubuntu 16.04 (Xenial).

**nginx**

Note: Please note that you **must** use Apache if you intend to use SSO using Shibboleth/SAML/ADFS. If that’s not the case because you do not use SSO at all or use e.g. OAuth, we recommend using nginx.

### 1. Install Packages

PostgreSQL and nginx are installed from their upstream repos to get much more recent versions.

```
apt install -y lsb-release wget gnupg
echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list
echo "deb http://nginx.org/packages/$(lsb_release -is | tr '[:upper:]' '[:lower:]')/ $(lsb_release -cs) nginx" > /etc/apt/sources.list.d/nginx.list
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add -
wget --quiet -O - https://nginx.org/keys/nginx_signing.key | apt-key add -
apt update
apt install -y --install-recomends postgresql-9.6 libpq-dev nginx python-dev python--virtualenv libxslt1-dev libxml2-dev libffi-dev libpcré3-dev libyaml-dev build--essential redis-server uwsgi uwsgi-plugin-python
```

If you use Debian, run this command:

```
apt install -y libjpeg62-turbo-dev
```

If you use Ubuntu, run this instead:

```
```
apt install -y libjpeg-turbo8-dev zlib1g-dev

Afterwards, make sure the services you just installed are running:

systemctl start postgresql.service redis-server.service

2. Create a Database

Let’s create a user and database for indico and enable the necessary Postgres extensions (which can only be done by the Postgres superuser).

su - postgres -c 'createuser indico'
su - postgres -c 'createdb -O indico indico'
su - postgres -c 'psql indico -c "CREATE EXTENSION unaccent; CREATE EXTENSION pg_trgm; ..."'

**Warning:** Do not forget to setup a cronjob that creates regular database backups once you start using Indico in production!

3. Configure uWSGI & nginx

The default uWSGI and nginx configuration files should work fine in most cases.

ln -s /etc/uwsgi/apps-available/indico.ini /etc/uwsgi/apps-enabled/indico.ini
cat > /etc/uwsgi/apps-available/indico.ini <<'EOF'
[uwsgi]
uid = indico
gid = nginx
umask = 027

processes = 4
enable-threads = true
chmod-socket = 770
chown-socket = indico:nginx
socket = /opt/indico/web/uwsgi.sock
stats = /opt/indico/web/uwsgi-stats.sock
protocol = uwsgi

master = true
auto-procname = true
procname-prefix-spaced = indico
disable-logging = true

plugin = python
single-interpreter = true

touch-reload = /opt/indico/web/indico.wsgi
wsgi-file = /opt/indico/web/indico.wsgi
virtualenv = /opt/indico/.venv

vacuum = true
buffer-size = 20480

EOF
memory-report = true
max-requests = 2500
harakiri = 900
harakiri-verbose = true
reload-on-rss = 2048
evil-reload-on-rss = 8192

Note: Replace \texttt{YOURHOSTNAME} in the next file with the hostname on which your Indico instance should be available, e.g. \texttt{indico.yourdomain.com}

```bash
cat > /etc/nginx/conf.d/indico.conf <<'EOF'
server {
    listen 80;
    listen [:]:80;
    server_name YOURHOSTNAME;
    return 301 https://$server_name$request_uri;
}

cat > /etc/nginx/conf.d/indico.conf <<'EOF'
server {
    listen 443 ssl http2;
    listen [:443] ssl http2 default ipv6only=on;
    server_name YOURHOSTNAME;

    ssl on;
    ssl_certificate /etc/ssl/indico/indico.crt;
    ssl_certificate_key /etc/ssl/indico/indico.key;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

    access_log /opt/indico/log/nginx/access.log combined;
    error_log /opt/indico/log/nginx/error.log;

    location /.xsf/indico/ {
        internal;
        alias /opt/indico/;
    }

    location ~ ^/(images|fonts)(.*)/(.+?)(__v[0-9a-f]+)?/\([^\+]+\)\$ { 
        alias /opt/indico/web/static/$1$2/$3.$5;
        access_log off;
    }

    (continues on next page)
```
4. Create an SSL Certificate

First, create the folders for the certificate/key and set restrictive permissions on them:

```
mkdir /etc/ssl/indico
chown root:root /etc/ssl/indico/
chmod 700 /etc/ssl/indico
```

If you are just trying out Indico you can simply use a self-signed certificate (your browser will show a warning which you will have to confirm when accessing your Indico instance for the first time).

**Note:** Do not forget to replace YOURHOSTNAME with the same value you used above

```
openssl req -x509 -nodes -newkey rsa:4096 -subj /CN=YOURHOSTNAME -keyout /etc/ssl/indico/indico.key -out /etc/ssl/indico/indico.crt
```

While a self-signed certificate works for testing, it is not suitable for a production system. You can either buy a certificate from any commercial certification authority or get a free one from Let's Encrypt.

**Note:** There’s an optional step later in this guide to get a certificate from Let’s Encrypt. We can’t do it right now since the nginx config references a directory yet to be created, which prevents nginx from starting.

5. Install Indico

Celery runs as a background daemon. Add a systemd unit file for it:
Now create a user that will be used to run Indico and switch to it:

```bash
useradd -rm -g nginx -d /opt/indico -s /bin/bash indico
su - indico
```

You are now ready to install Indico:

```bash
virtualenv ~/.venv
source ~/.venv/bin/activate
pip install -U pip setuptools
pip install indico
```

6. Configure Indico

Once Indico is installed, you can run the configuration wizard. You can keep the defaults for most options, but make sure to use `https://YOURHOSTNAME` when prompted for the Indico URL. Also specify valid email addresses when asked and enter a valid SMTP server Indico can use to send emails. When asked for the default timezone make sure this is the main time zone used in your Indico instance.

```
indico setup wizard
```

Now finish setting up the directory structure and permissions:

```bash
mkdir ~/log/nginx
cmpgo-rwx ~/ -/.* -./[^.]*
chmog 710 ~/archive ~/cache ~/log ~/tmp
chmog 750 ~/web ~/venv
chmog g+w ~/log/nginx
echo -e "\nSTATIC_FILE_METHOD = ('xaccelredirect', {'/opt/indico': '/.xsf/indico'})" > ~/etc/indico.conf
```
7. Create database schema

Finally, you can create the database schema and switch back to root:

```
indico db prepare
exit
```

8. Launch Indico

You can now start Indico and set it up to start automatically when the server is rebooted:

```
systemctl restart uwsgi.service nginx.service indico-celery.service
systemctl enable uwsgi.service nginx.service postgresql.service redis-server.service
˓
→ indico-celery.service
```

9. Optional: Get a Certificate from Let’s Encrypt

**Note:** You need to use at least Debian 9 (Stretch) to use certbot. If you are still using Debian 8 (Jessie), consider updating or install certbot from backports.

If you use Ubuntu, install the certbot PPA:

```
apt install -y software-properties-common
add-apt-repository -y ppa:certbot/certbot
apt update
```

To avoid ugly SSL warnings in your browsers, the easiest option is to get a free certificate from Let’s Encrypt. We also enable the cronjob to renew it automatically:

```
apt install -y python-certbot-nginx
certbot --nginx --rsa-key-size 4096 --no-redirect --staple-ocsp -d YOURHOSTNAME
rm -rf /etc/ssl/indico
systemctl start certbot.timer
systemctl enable certbot.timer
```

10. Create an Indico user

Access https://YOURHOSTNAME in your browser and follow the steps displayed there to create your initial user.

11. Install TeXLive

Follow the *LaTeX install guide* to install TeXLive so Indico can generate PDF files in various places.

Apache

1. Install Packages

PostgreSQL is installed from its upstream repos to get a much more recent version.
apt install -y lsb-release wget gnupg
echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add -
apt update
apt install -y --install-recommends postgresql-9.6 libpq-dev apache2 libapache2-mod-proxy-uwsgi libapache2-mod-xsendfile python-dev python-virtualenv libxml2-dev libffi-dev libpcre3-dev libyaml-dev build-essential redis-server uwsgi uwsgi-plugin-python

If you use Debian, run this command:
```bash
apt install -y libjpeg62-turbo-dev
```

If you use Ubuntu, run this instead:
```bash
apt install -y libjpeg-turbo8-dev zlib1g-dev
```

Afterwards, make sure the services you just installed are running:
```bash
systemctl start postgresql.service redis-server.service
```

### 2. Create a Database

Let’s create a user and database for indico and enable the necessary Postgres extensions (which can only be done by the Postgres superuser).

```bash
su - postgres -c 'createuser indico'
su - postgres -c 'createdb -O indico indico'
su - postgres -c 'psql indico -c "CREATE EXTENSION unaccent; CREATE EXTENSION pg_trgm;"'
```

**Warning:** Do not forget to setup a cronjob that creates regular database backups once you start using Indico in production!

### 3. Configure uWSGI & Apache

The default uWSGI and Apache configuration files should work fine in most cases.

```bash
ln -s /etc/uwsgi/apps-available/indico.ini /etc/uwsgi/apps-enabled/indico.ini
```

```bash
cat > /etc/uwsgi/apps-available/indico.ini <<'EOF'
[uwsgi]
uid = indico
gid = www-data
umask = 027

processes = 4
enable-threads = true
socket = 127.0.0.1:8008
stats = /opt/indico/web/uwsgi-stats.sock
protocol = uwsgi
EOF
```

(continues on next page)
master = true
auto-procname = true
procname-prefix-spaced = indico
disable logging = true

plugin = python
single-interpreter = true
touch-reload = /opt/indico/web/indico.wsgi
wsgi-file = /opt/indico/web/indico.wsgi
virtualenv = /opt/indico/.venv

vacuum = true
buffer-size = 20480
memory-report = true
max-requests = 2500
harakiri = 900
harakiri-verbose = true
reload-on-rss = 2048
evil-reload-on-rss = 8192
EOF

Note: Replace YOURHOSTNAME in the next files with the hostname on which your Indico instance should be available, e.g. indico.yourdomain.com

```bash
cat > /etc/apache2/sites-available/indico-sslredirect.conf <<'EOF'
<VirtualHost *:80>
  ServerName YOURHOSTNAME
  RewriteEngine On
  RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>
EOF

cat > /etc/apache2/sites-available/indico.conf <<'EOF'
<VirtualHost *:443>
  ServerName YOURHOSTNAME
  DocumentRoot "/var/empty/apache"
  SSLEngine on
  SSLCertificateFile /etc/ssl/indico/indico.crt
  SSLCertificateKeyFile /etc/ssl/indico/indico.key
  SSLProtocol all -SSLv2 -SSLv3
  SSLCipherSuite ECDHE-ECDSA-CHACHA20-POLY1305:ECDSA-CHACHA20-POLY1305:::ALL
  SSLHonorCipherOrder on
  XSendFile on
</VirtualHost>
EOF
```
XSendFilePath /opt/indico
CustomLog /opt/indico/log/apache/access.log combined
ErrorLog /opt/indico/log/apache/error.log
LogLevel error
ServerSignature Off

AliasMatch "^/(images|fonts)/(.*)(__v[0-9a-f]+)?\.([^.]*)$" " /opt/indico/web/
static/$1$2/$3.$5"
AliasMatch "^/(css|dist|images|fonts)/(.*);$" " /opt/indico/web/static/$1/$2"
Alias /robots.txt /opt/indico/web/static/robots.txt

SetEnv UWsgi_Scheme https
ProxyPass / uwsgi://127.0.0.1:8008/

<Directory /opt/indico>
AllowOverride None
Require all granted
</Directory>
</VirtualHost>
EOF

Now enable the necessary modules and the indico site in apache:

```bash
a2enmod proxy_uwsgi rewrite ssl xsendfile
da2disite 000-default
da2ensite indico indico-sslredir
```

### 4. Create an SSL Certificate

First, create the folders for the certificate/key and set restrictive permissions on them:

```bash
mkdir /etc/ssl/indico
chown root:root /etc/ssl/indico/
chmod 700 /etc/ssl/indico
```

If you are just trying out Indico you can simply use a self-signed certificate (your browser will show a warning which you will have to confirm when accessing your Indico instance for the first time).

**Note:** Do not forget to replace YOURHOSTNAME with the same value you used above

```bash
openssl req -x509 -nodes -newkey rsa:4096 -subj /CN=YOURHOSTNAME -keyout /etc/ssl/
-indico/indico.key -out /etc/ssl/indico/indico.crt
```

While a self-signed certificate works for testing, it is not suitable for a production system. You can either buy a certificate from any commercial certification authority or get a free one from Let’s Encrypt.

**Note:** There’s an optional step later in this guide to get a certificate from Let’s Encrypt. We can’t do it right now since the Apache config references a directory yet to be created, which prevents Apache from starting.
5. Install Indico

Celery runs as a background daemon. Add a systemd unit file for it:

```bash
cat > /etc/systemd/system/indico-celery.service <<'EOF'
[Unit]
Description=Indico Celery
After=network.target

[Service]
ExecStart=/opt/indico/.venv/bin/indico celery worker -B
Restart=always
SyslogIdentifier=indico-celery
User=indico
Group=www-data
UMask=0027
Type=simple
KillMode=mixed
TimeoutStopSec=300

[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload
```

Now create a user that will be used to run Indico and switch to it:

```bash
useradd -rm -g www-data -d /opt/indico -s /bin/bash indico
su - indico
```

You are now ready to install Indico:

```bash
Note: If you need to migrate from Indico 1.2, you must install Indico 2.0, regardless of what the latest Indico version is. If this is the case for you, replace the last command in the block below with `pip install 'indico<2.1'
```

```bash
virtualenv ~/.venv
source ~/.venv/bin/activate
pip install -U pip setuptools
pip install indico
```

6. Configure Indico

Once Indico is installed, you can run the configuration wizard. You can keep the defaults for most options, but make sure to use `https://YOURHOSTNAME` when prompted for the Indico URL. Also specify valid email addresses when asked and enter a valid SMTP server Indico can use to send emails. When asked for the default timezone make sure this is the main time zone used in your Indico instance.

```bash
indico setup wizard
```

Now finish setting up the directory structure and permissions:

```bash
mkdir ~/log/apache
chmod go-rwx ~/* ~/.[^.]*
```

(continues on next page)
6. Configure file permissions

Finally, you can create the database schema and switch back to root:

```bash
indico db prepare
exit
```

8. Launch Indico

You can now start Indico and set it up to start automatically when the server is rebooted:

```bash
systemctl restart uwsgi.service apache2.service indico-celery.service
systemctl enable uwsgi.service apache2.service postgresql.service redis-server.
```

9. Optional: Get a Certificate from Let's Encrypt

**Note:** You need to use at least Debian 9 (Stretch) to use certbot. If you are still using Debian 8 (Jessie), consider updating or install certbot from backports.

If you use Ubuntu, install the certbot PPA:

```bash
apt install -y software-properties-common
add-apt-repository -y ppa:certbot/certbot
apt update
```

To avoid ugly SSL warnings in your browsers, the easiest option is to get a free certificate from Let’s Encrypt. We also enable the cronjob to renew it automatically:

```bash
apt install -y python-certbot-apache
certbot --apache --rsa-key-size 4096 --no-redirect --staple-ocsp -d YOURHOSTNAME
rm -rf /etc/ssl/indico
systemctl start certbot.timer
systemctl enable certbot.timer
```

10. Create an Indico user

Access [https://YOURHOSTNAME](https://YOURHOSTNAME) in your browser and follow the steps displayed there to create your initial user.

11. Install TeXLive

Follow the [LaTeX install guide](https://www.latex-project.org/install/) to install TeXLive so Indico can generate PDF files in various places.

1.1. Installation guides
Optional: Shibboleth

If your organization uses Shibboleth/SAML-based SSO, follow these steps to use it in Indico:

1. Install Shibboleth

```
apt install -y libapache2-mod-shib2
a2enmod shib2
```

2. Configure Shibboleth

This is outside the scope of this documentation and depends on your environment (Shibboleth, SAML, ADFS, etc). Please contact whoever runs your SSO infrastructure if you need assistance.

3. Enable Shibboleth in Apache

Add the following code to your `/etc/apache2/sites-available/indico.conf` right before the `AliasMatch` lines:

```
<LocationMatch "^(/Shibboleth\.sso|/login/shib-sso/shibboleth)"
   AuthType shibboleth
   ShibRequestSetting requireSession 1
   ShibExportAssertion Off
   Require valid-user
</LocationMatch>
```

4. Enable Shibboleth in Indico

Add the following code to your `/opt/indico/etc/indico.conf`:

```
# SSO
AUTH_PROVIDERS = {
    'shib-sso': {
        'type': 'shibboleth',
        'title': 'SSO',
        'attrs_prefix': 'ADFS_',
        'callback_uri': '/login/shib-sso/shibboleth',
        # 'logout_uri': 'https://login.yourcompany.tld/logout'
    }
}
IDENTITY_PROVIDERS = {
    'shib-sso': {
        'type': 'shibboleth',
        'title': 'SSO',
        'identifier_field': 'ADFS_LOGIN',
        'mapping': {
            'affiliation': 'ADFS_HOMEINSTITUTE',
            'first_name': 'ADFS_FIRSTNAME',
            'last_name': 'ADFS_LASTNAME',
            'email': 'ADFS_EMAIL',
        }
    }
}
```

(continues on next page)
The values for `attrs_prefix`, `mapping` and `identifier_field` may be different in your environment. Uncomment and set `logout_uri` if your SSO infrastructure provides a logout URL (usually used to log you out from all applications).

If you only want to use SSO, without allowing people to login locally using username/password, disable it by setting `LOCAL_IDENTITIES = False` in `indico.conf`.

**Warning:** We assume that emails received from SSO are already validated. If this is not the case, make sure to disable `trusted_email` which will require email validation in Indico when logging in for the first time. Otherwise people could take over the account of someone else by using their email address!

**Note:** The example config is rather simple and only accesses data from SSO during login. This is not sufficient for advanced features such as automatic synchronization of names, affiliations and phone numbers or using centrally managed groups. To use these features, you need to use e.g. the LDAP identity provider and use the information received via SSO to retrieve the user details from LDAP. If you need assistance with this, feel free to ask us on IRC (#indico @ Freenode) or via e-mail (indico-team@cern.ch).

**Note:** Please note that you **must** use Apache if you intend to use SSO using Shibboleth/SAML/ADFS. If that’s not the case because you do not use SSO at all or use e.g. OAuth, we recommend using nginx.

### 1.1.2 Upgrade

It is important to keep your Indico instance up to date to have the latest bug fixes and features. Upgrading can be done with almost no user-facing downtime.

**Warning:** When upgrading a production system it is highly recommended to create a database backup before starting.

First of all, stop the Celery worker. To do so, run this as `root`:

```
systemctl stop indico-celery.service
```

Now switch to the `indico` user and activate the virtualenv:

```
su - indico
source ~/.venv/bin/activate
```

If you are on CentOS, update your PATH to avoid errors in case the new Indico version needs to install an updated version of the PostgreSQL client library (psycopg2):
You are now ready to install the latest version of Indico:

```bash
pip install -U indico
```

If you installed the official plugins, update them too:

```bash
pip install -U indico-plugins
```

Some versions may include database schema upgrades. Make sure to perform them immediately after upgrading. If there are no schema changes, the command will simply do nothing.

```bash
indico db upgrade
indico db --all-plugins upgrade
```

**Note:** Some database structure changes require an exclusive lock on some tables in the database. Unless you have very high activity on your instance, this lock can be acquired quickly, but if the upgrade command seems to hang for more than a few seconds, you can restart uWSGI by running `systemctl restart uwsgi.service` as `root` (in a separate shell, i.e. don’t abort the upgrade command!) which will ensure nothing is accessing Indico for a moment.

Unless you just restarted uWSGI, it is now time to reload it so the new version is actually used:

```bash
touch ~/web/indico.wsgi
```

Also start the Celery worker again (once again, as `root`):

```bash
systemctl start indico-celery.service
```

### Upgrading from 2.x to 2.2

**Warning:** Keep in mind that running Indico from a subdirectory such as `https://example.com/indico` is no longer supported by the packages we provide on PyPI. Please use a subdomain instead.

When updating to version 2.2 you need to perform some extra steps due to the changes in Indico’s static asset pipeline. After installing 2.2, run `indico setup create-symlinks` (still as the `indico` user) to create the new symlink.

You can also perform some clean-up:

```bash
rm /opt/indico/web/htdocs
rm -rf /opt/indico/assets
sed -i -e '/ASSETS_DIR/d' ~/etc/indico.conf
```

Now switch back to `root` and update the webserver config as explained below.

### Apache

Open `/etc/httpd/conf.d/indico.conf` (CentOS) or `/etc/apache2/sites-available/indico.conf` (Debian) with an editor and replace this snippet:
with this one:

```
AliasMatch "^/(images|fonts)(.*)/(.+?)(__v[0-9a-f]+)?\.(\[^.]+)$" "/opt/indico/web/static/$1$2/$3.$5"
```

Reload apache using systemctl reload apache2.service.

nginx

Open `/etc/nginx/conf.d/indico.conf` with an editor and replace this snippet:

```
location ~ ^/static/assets/(core|(?!=plugin|theme)-[^/]+)/(.*$) 
  { 
    alias /opt/indico/assets/$1/$2; 
    access_log off; 
  }

location ~ ^/(css|images|js|static(?!/plugins|/assets|/themes|/custom))(.*)$ 
  { 
    alias /opt/indico/web/htdocs/$1$2; 
    access_log off; 
  }

location /robots.txt 
  { 
    alias /opt/indico/web/htdocs/robots.txt; 
    access_log off; 
  }
```

with this one:

```
location ~ ^/(images|fonts)(.*)/(.+?)(__v[0-9a-f]+)?\.(\[^.]+)$ 
  { 
    alias /opt/indico/web/static/$1$2/$3.$5; 
    access_log off; 
  }

location ~ ^/(css|dist|images|fonts)/(.*$) 
  { 
    alias /opt/indico/web/static/$1/$2; 
    access_log off; 
  }

location /robots.txt 
  { 
    alias /opt/indico/web/static/robots.txt; 
    access_log off; 
  }
```

Reload nginx using systemctl reload nginx.service.

If you are using customizations using the `CUSTOMIZATION_DIR` setting, see its updated documentation as you will have to update those customizations.
Upgrading from 1.9.11 to 2.0

Make sure that you have the latest 1.9.11 version installed and that you used `indico db upgrade` to have the most recent database structure.

First of all, if you had installed any plugins manually, you need to uninstall them first as we changed some of the Python distribution names so if you do not uninstall them, you will get errors about duplicate plugins.

```
pip freeze | grep -Po 'indico(?!-fonts).+(?===)' | pip uninstall -y
```

**Note:** If you used `pip install -e` to install the plugins, the command above will not work and you need to manually uninstall them. All the plugin packages have names like `indico_chat` or `indico_payment_manual`. If you are unsure about what to uninstall here, please contact us.

To upgrade to 2.0, follow the upgrade instructions above, but skip the DB upgrade commands. After successfully running the upgrade, use `indico db reset_alembic` to clear pre-2.0 database migration information, since all the old migration steps from the 1.9.x version line have been removed in 2.0.

The names of all settings changed in 2.0; instead of using CamelCased names they now use UPPER_SNAKE_CASE. The old names still work, but we recommend updating the config file anyway. You can find a list of all the new option names in the code. Most renames are pretty straightforward; only the following options have been changed in more than just capitalization:

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFLatexProgram</td>
<td>XELATEX_PATH</td>
</tr>
<tr>
<td>IsRoomBookingActive</td>
<td>ENABLE_ROOMBOOKING</td>
</tr>
<tr>
<td>SanitizationLevel</td>
<td>removed</td>
</tr>
</tbody>
</table>

The format of the logging config changed. The old file `/opt/indico/etc/logging.conf` is not used anymore and can be deleted. Run `indico setup create-logging-config /opt/indico/etc/` to create the new `logging.yaml` which can then be customized if needed.

1.1.3 Upgrade Indico from 1.2

If you’re running a version that is lower than 2.0, you will have to run a special migration command provided by the `indico-migrate` package. This document will guide you over the steps needed to perform the upgrade.

**Prerequisites**

In order to migrate to version 2.0 of Indico you will first of all need to make sure you have at least version 1.2 of Indico installed. Migration of databases using earlier versions will either fail or very likely result in data loss. So, please make sure that you are on 1.2.x before migrating.

**Warning:** If you are running a version of the experimental (thus unsupported) 1.9.x branch, you will have to perform a step-by-step migration. We hope that, as advised, no-one upgraded to intermediate 1.9.x releases. If you did and need help with it, please ping us on IRC.

**Backing up ZODB**

The migration script doesn’t write to the ZODB, but we still recommend that you make a backup just in case:
You should replace `<some-place-safe>` with the directory in your filesystem where you want to keep the backup. As for `<indico-db-dir>`, that’s the directory where the database file is kept. That should be `/opt/indico/db` in most Indico installations.

Make sure that backup files have been created (you should have an `*.index` and an `*.fs` file).

Now, let’s shut down the ZEO daemon:

```
  zdaemon -C /opt/indico/etc/zdctl.conf stop
```

Double check that the daemon is not running:

```
  zdaemon -C /opt/indico/etc/zdctl.conf status
```

## Moving legacy data

Indico 2.0 will use a directory structure that is similar to Indico 1.x, so first of all you will need to rename the old tree:

```
  mv /opt/indico /opt/indico-legacy
```

**Warning:** After the migration is done, do not delete the `/opt/indico-legacy` directory without first moving the archive dir elsewhere. Please read the full guide until the end.

## Installing Indico 2.0

The first step should be to have a working Indico 2.0 setup. In order to do that, you should follow the regular Indico 2.x installation instructions up to the “Configure Indico” step. We provide below direct links to the relevant sections of the installation guides.

On a **Debian/Ubuntu** system:

<table>
<thead>
<tr>
<th>nginx</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Install Packages</td>
<td>1. Install Packages</td>
</tr>
<tr>
<td>2. Create a Database</td>
<td>2. Create a Database</td>
</tr>
<tr>
<td>3. Configure uWSGI &amp; nginx</td>
<td>3. Configure uWSGI &amp; Apache</td>
</tr>
<tr>
<td>5. Install Indico</td>
<td>5. Install Indico</td>
</tr>
<tr>
<td>6. Configure Indico</td>
<td>6. Configure Indico</td>
</tr>
</tbody>
</table>

On a **CentOS7-based** system:
Configuration Wizard

You will then need to run the Configuration Wizard, following the normal installation guide (Debian/Ubuntu or CentOS). When the wizard asks you about the “Old archive dir”, make sure to set it to the archive dir in the indico-legacy directory.

... If you are upgrading from Indico 1.2, please specify the path to the ArchiveDir of the old indico version. Leave this empty if you are not upgrading.
Old archive dir: /opt/indico-legacy/archive
...

Running `indico-migrate`

First of all, make sure that you are using the user and virtualenv created using the step “Install Indico” and that the legacy dir is owned by this user:

```
chown -R indico /opt/indico-legacy
su - indico
source ~/.venv/bin/activate
```

You should then install the package using:

```
pip install indico-migrate
```

`indico-migrate` requires a series of parameters that have to be tuned according to your current setup. We now provide a list of values that should work in most standard Indico installations. However, please carefully read the documentation of the indico-migrate command, to make sure there are no option conflicts with your setup.

Most frequently, `indico-migrate` will work, followed by the following parameters:

- `--archive-dir /opt/indico-legacy/archive`
- `--storage-backend legacy`
- `--default-email default@<organization-hostname>`
- `--default-currency EUR`
- `--symlink-target ~/archive/legacy_symlinks/`
- `--symlink-backend legacy-symlinks`
• **--migrate-broken-events** (optional - use it if you want to migrate events that don’t belong to any category in v1.2. If any such events exist, the will be added to a new category named *Lost & Found*.

(don’t forget to replace `<organization-hostname>` with the e-mail hostname of your organization)

An example:

```
indico-migrate postgresql:///indico file:///opt/indico-legacy/db/Data.fs --archive-dir /opt/indico-legacy/archive --storage-backend legacy --default-email <default@acme.example.com> --default-currency EUR --symlink-target ~/archive/legacy --symlinks --symlink-backend legacy-symlinks --migrate-broken-events
```

**Note:** If for some reason the migration fails, `indico-migrate` will ask you whether you would like to post an error report on a public pastebin (Gist). The link will not be advertised and only the log information that was shown on screen (plus the exception traceback that was printed) will be included. If you are not comfortable with letting `indico-migrate` post this on a public pastebin, you can always send us your `migration.log` file (which gets generated automatically).

---

### Post-migration work

After the migration is done you may need to apply some adjustments in your `indico.conf`. You may want to read our guide on how to configure an Identity/Authentication provider.

We really recommend as well that you move your old Indico archive (`/opt/indico-legacy/archive`) inside your new Indico directory:

```
mv /opt/indico-legacy/archive /opt/indico/legacy-archive
```

The legacy archive will remain **read-only**. You should update your `indico.conf` (STORAGE_BACKENDS option) to reflect the new path:

```
STORAGE_BACKENDS = {
    # ...
    'legacy': 'fs-readonly:/opt/indico/legacy-archive'
    # ...
}
```

---

### Finishing up

You can now proceed with the remaining installation steps:

On a **Debian/Ubuntu** system:

<table>
<thead>
<tr>
<th>8. Launch Indico</th>
<th>8. Launch Indico</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. Create an Indico user</td>
<td>10. Create an Indico user</td>
</tr>
<tr>
<td>11. Install TeXLive</td>
<td>11. Install TeXLive</td>
</tr>
</tbody>
</table>

On a **CentOS7-based** system:

---

1.1. Installation guides 35
**Sanitizing HTML**

Indico 2.0 uses Markdown for the descriptions of contributions and categories. Contribution descriptions that previously contained HTML will still work, but new ones will only support Markdown syntax (including basic HTML). As for the descriptions of categories, they are interpreted as Markdown as of version 2.0, which means that some existing data may be broken. In order to make the lives of users who are migrating easier, we have included with `indico-migrate` a command that automatically performs the migration of Category descriptions to Markdown.

First of all, let’s see what would be the impact of running the command:

```bash
indico-html-sanitize --dry-run -v -l log.html category_descriptions
```

By opening `log.html` you will be able to check if there are any special cases that will need manual intervention. If you’re happy with the changes, you can just choose to save them:

```bash
indico-html-sanitize category_descriptions
```

**Removing old data**

Even if you’re sure the migration succeeded and all data was kept, please keep around the backup of your ZODB you made at the beginning of this guide. **After and only after** having moved the legacy archive to the new Indico dir and stored a backup of your ZODB in a safe place, you can proceed to delete the old `/opt/indico` directory:

```bash
rm -rf /opt/indico-legacy
```

### 1.1.4 Installation guide (development)

**Installing System Packages**

Web assets such as JavaScript and SCSS files are compiled using Webpack, which requires NodeJS to be present. You can find information on how to install NodeJS here.

Do not use the default NodeJS packages from your Linux distribution as they are usually outdated or come with an outdated npm version.

**CentOS/Fedora**

```bash
yum install -y gcc redis python-devel python-virtualenv libjpeg-turbo-devel libxslt-devel libffi-devel pcre-devel libyaml-devel redhat-rpm-config 
libxml2-devel postgresql postgresql-server postgresql-contrib libpq-devel systemctl start redis.service postgresql.service
```
Debian/Ubuntu

```
apt install -y --install-recommends python-dev python-virtualenv libxslt1-dev libxml2-dev libffi-dev libpcre3-dev libyaml-dev build-essential redis-server postgresql libpq-dev
```

Then on Debian:
```
apt install -y libjpeg62-turbo-dev
```

And on Ubuntu:
```
apt install -y libjpeg-turbo8-dev zlib1g-dev
```

macOS

We recommend that you use Homebrew:

```
brew install python2 redis libjpeg libffi pcre libyaml postgresql
brew services start postgresql
pip install virtualenv
```

Note: Homebrew dropped support for the python2 formula at the end of 2019. As an alternative you can install it directly using the latest commit:

```
brew install https://raw.githubusercontent.com/Homebrew/homebrew-core/86a44a0a552c673a05f11018459c9f5faae3becc/Formula/python@2.rb
```

Creating the directory structure

You will need a directory in your file system to store Indico as well as its data files (archives, etc...). Some developers keep all their code inside a dev or code dir. We will assume dev here.
```
mkdir -p ~/dev/indico/data
```

We will need a virtualenv where to run Indico:
```
cd ~/dev/indico
virtualenv env -p /usr/bin/python2.7
```

Cloning Indico

First, let’s clone Indico’s code base. If you’re going to contribute back to the project, it’s probably best if you clone your own GitHub fork of the project and set it as the origin:

```
git clone git@github.com:<your-github-username>/indico.git src
cd src
git remote add upstream https://github.com/indico/indico.git
cd ..
```

Otherwise, cloning the upstream repository as the origin should be enough:
git clone https://github.com/indico/indico.git src

If you’re going to be changing the standard Indico plugins and/or the documentation, you can also clone those:
mkdir plugins
git clone https://github.com/indico/indico-plugins.git plugins/base
git clone https://github.com/indico/indico-user-docs.git user-docs

Setting up Maildump (recommended)

Some actions in Indico trigger automatic e-mails. Those will normally have to be routed through an SMTP server. This can become a problem if you’re using production data and/or real e-mails, as users may end up being spammed unnecessarily. This is why we advise that you include a fake SMTP server in your development setup. Maildump does exactly this and runs on Python. It should be quite simple to set up:

code
virtualenv maildump -p /usr/bin/python2.7
./maildump/bin/pip install -U pip setuptools
./maildump/bin/pip install maildump
./maildump/bin/maildump -p /tmp/maildump.pid

done

You’ll then be able to access the message log at http://localhost:1080.

Creating the DB

Let’s get into the Indico virtualenv:

```
sudo -u postgres createuser $USER --createdb
sudo -u postgres createdb indico_template -O $USER
sudo -u postgres psql indico_template -c "CREATE EXTENSION unaccent; CREATE EXTENSION pg_trgm;"
createdb indico -T indico_template
```

Configuring

Let’s get into the Indico virtualenv:

```
sudo -u postgres createuser $USER --createdb
sudo -u postgres createdb indico_template -O $USER
sudo -u postgres psql indico_template -c "CREATE EXTENSION unaccent; CREATE EXTENSION pg_trgm;"
createdb indico -T indico_template
```

```
source ./env/bin/activate
pip install -U pip setuptools

cd src
pip install -r requirements.dev.txt
pip install -e .
npm install
```

Then, follow the instructions given by the wizard:

```
indico setup wizard --dev
```

You can then initialize the DB:

```
indico db prepare
```
Running Indico

You will need two shells running in parallel. The first one will run the webpack watcher, which compiles the JavaScript and style assets every time you change them:

```
./bin/maintenance/build-assets.py indico --dev --watch
```

On the second one we’ll run the Indico Development server:

```
indico run -h <your-hostname> -q --enable-evalex
```

Double-check that your hostname matches that which has been set in the config file (by the wizard).

It is also worth mentioning that when working on a plugin, it is necessary to run another webpack watcher to build the plugin assets. That can be accomplished using the same command as above with an argument specifying which plugin you want to build the assets for:

```
./bin/maintenance/build-assets.py <plugin-name> --dev --watch
```

You can also build the assets for all the plugins:

```
./bin/maintenance/build-assets.py all-plugins --dev <plugins-directory>
```

Installing TeXLive (optional)

If you need PDF generation in certain parts of Indico to work (e.g. for contributions and the Book of Abstracts), you need LaTeX. To install it, follow the [LaTeX install guide](#).

Using HTTPS through nginx (optional)

If you wish to open your development server to others, then we highly recommend that you properly set HTTPS. While you could do so directly at the development server, it’s normally easier to proxy it through nginx and have it serve static files as well.

You should obviously install nginx first:

```
sudo yum install nginx  # centos/fedora
sudo apt install nginx  # debian/ubuntu
brew install nginx     # macOS
```

Here is an example of a `nginx.conf` you can use. It assumes your username is jdoe and the hostname is acme.example.org:

```
user jdoe users;
worker_processes 4;
error_log /var/log/nginx/error.log info;
pid /run/nginx.pid;

events {
    worker_connections 1024;
    use epoll;
}

http {
    access_log off;

    (continues on next page)
```
sendfile on;
tcp_nopush on;
tcp_nodelay on;

keepalive_timeout 75 20;
types_hash_max_size 2048;
ignore_invalid_headers on;

connection_pool_size 256;
client_header_buffer_size 10k;
large_client_header_buffers 4 20k;
request_pool_size 4k;
client_max_body_size 2048m;

proxy_buffers 32 32k;
proxy_buffer_size 32k;
proxy_busy_buffers_size 128k;

gzip on;
gzip_min_length 1100;
gzip_buffers 4 8k;
gzip_types text/plain text/css application/x-javascript;

include /etc/nginx/mime.types;
default_type application/octet-stream;

server {
    listen [::]:80 ipv6only=off;
    server_name acme.example.org;

    access_log /var/log/nginx/acme.access_log combined;
    error_log /var/log/nginx/acme.error_log info;

    root /var/empty;

    return 302 https://$server_name$request_uri;
}

server {
    listen [::]:443 ipv6only=off http2;
    server_name acme.example.org;

    ssl on;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_certificate /home/jdoe/acme.crt;
    ssl_certificate_key /home/jdoe/acme.key;

    access_log /var/log/nginx/acme.ssl_access_log combined;
    error_log /var/log/nginx/acme.ssl_error_log info;

    root /var/empty;
}
This configuration also assumes you’ve already got a secret key and certificate stored in `~/acme.key` and `acme.crt` respectively. In most cases you will probably use a self-signed certificate. There are many guides on-line on how to generate a self-signed certificate, so we will not cover it here.

If you’re using SELinux, you will need to set the following configuration options:

```
sudo setsebool -P httpd_can_network_connect 1
sudo setsebool -P httpd_read_user_content 1
```

Uploading large files will probably fail unless you do:

```
sudo chown -R jdoe:nginx /var/lib/nginx/tmp/
```

The Indico dev server should be run with the `--proxy` option:

```
indico run -h 127.0.0.1 -p 8000 -q --enable-evalex --url https://acme.example.org --proxy
```

You can then start nginx and access `https://acme.example.org` directly.

### 1.1.5 Plugins

We provide a meta-package that contains all official plugins. Before installing it, make sure you are logged in as the `indico` user and inside the Indico environment:

```
su - indico
source ~/.venv/bin/activate
```

Now install the package which will automatically install our plugins:

```
pip install indico-plugins
```

**Note:** Having all plugins installed has no disadvantages; only plugins enabled in `indico.conf` are actually loaded and executed. If you do not use the `indico-plugins` package, we won’t be able to display a notification when updates are available and you would have to update all the plugins separately.
You can use the `indico setup list-plugins` command to see which plugins are installed and which name to use in the config file to load them.

To enable plugins, add a `PLUGINS` entry to `/opt/indico/etc/indico.conf`. For example, the following line would enable the “Bank Transfer” and “PayPal” payment plugins:

```
PLUGINS = {'payment_manual', 'payment_paypal'}
```

Some plugins contain additional database tables. Run the plugin database migrations to create them (if you do not have any plugins with custom tables, the command will simply do nothing):

```
indico db --all-plugins upgrade
```

After any change to the config file, you need to reload uWSGI:

```
touch ~/web/indico.wsgi
```

It is also a good idea to restart the Celery worker (as `root`) since some plugins may come with background tasks:

```
systemctl restart indico-celery.service
```

### 1.1.6 LaTeX

Indico uses LaTeX (xelatex to be exact) to generate some PDF files such as the *Book of Abstracts* and the PDF versions of contributions. If you do not need these features, you can skip this part of the documentation and avoid installing LaTeX altogether.

Since Indico requires quite a few LaTeX packages which are not always installed by default when using the `texlive` packages of the various linux distributions, we recommend installing it manually.

First of all, you will need to install some dependencies so that all TeX formats are generated successfully upon TeXLive installation.

```
yum install fontconfig ghostscript  # CentOS / CC7
apt install libfontconfig1 ghostscript # Debian / Ubuntu
```

You are now ready to install TeXLive. The following commands should work fine to install everything you need. You need to run the installation as root or create `/opt/texlive` as root and grant your user write access to it.

Download the installer and cd to its location (the directory name contains the date when the package was built, so use the wildcard or type the name manually based on the output when unpacking the archive):

```
cd /tmp
wget http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
tar xvzf install-tl-unx.tar.gz
cd install-tl-*/
```

Create the setup config file to install all the packages you need:

```
cat > texlive.profile <<'EOF'
selected_scheme scheme-full
TEXDIR /opt/texlive
TEXMFCONFIG ~/.texlive/texmf-config
TEXMFHOME ~/texmf
TEXMFSYSCONFIG /opt/texlive/texmf-config
TEXMFSYSVAR /opt/texlive/texmf-var
EOF
```

(continues on next page)
Start the installer and wait for it to complete. This may take between a few minutes and a few hours depending on the speed of the (randomly chosen) mirror.

```
./install-tl --profile texlive.profile
```

After installing it, add this line to your `indico.conf` file to use your new TeXLive installation:

```
XELATEX_PATH = '/opt/texlive/bin/x86_64-linux/xelatex'
```

If you are in a production setup, reload uWSGI using `touch /opt/indico/web/indico.wsgi` to reload the config file.

As security-related updates are released frequently, it is also a good idea to periodically update the TeXLive packages by running:

```
/opt/texlive/bin/x86_64-linux/tlmgr update --self --all
```
Indico is very flexible and many things can be configured/customized in its configuration file.

## 2.1 Configuration

Indico is very flexible and many things can be configured/customized in its configuration file.

### 2.1.1 Settings

`indico.conf` is Indico’s main configuration file. Its initial version is usually generated when running `indico setup wizard` as described in the Installation Guide, but depending on the setup it should be modified later.

The config file is loaded from the path specified in the `INDICO_CONFIG` environment variable; if no such path is set, the config file (or a symlink to it) is searched in the following places, in order:

- `<indico_package_path>/indico.conf` (development setups only)
- `~/.indico.conf`
- `/etc/indico.conf`

The file is executed as a Python module, so anything that is valid Python 2.7 code can be used in it. When defining temporary variables that are not config options, their name should be prefixed with an underscore; otherwise you will get a warning about unknowing config options being defined.

### Authentication

**LOCAL_IDENTITIES**

This setting controls whether local Indico accounts are available. If no centralized authentication infrastructure (e.g. LDAP, OAuth, or another kind of SSO) is used, local accounts are the only way of logging in to Indico.

*Default: True*
**LOCAL_GROUPS**

This setting controls whether local Indico groups are available. If no centralized authentication infrastructure that supports groups (e.g. LDAP) is used, local groups are the only way to define groups in Indico, but if you do have central groups it may be useful to disable local ones to have all groups in one central place.

Default: True

**LOCAL_REGISTRATION**

This setting controls whether people accessing Indico can create a new account. Admins can always create new local accounts, regardless of this setting.

This setting is only taken into account if `LOCAL_IDENTITIES` are enabled.

Default: True

**LOCAL_MODERATION**

This setting controls whether a new registration needs to be approved by an admin before the account is actually created.

This setting is only taken into account if `LOCAL_IDENTITIES` and `LOCAL_REGISTRATION` are enabled.

Default: False

**EXTERNAL_REGISTRATION_URL**

The URL to an external page where people can register an account that can then be used to login to Indico (usually via LDAP/SSO).

This setting is only taken into account if `LOCAL_IDENTITIES` are disabled.

Default: None

**AUTH_PROVIDERS**

A dict defining Flask-Multipass authentication providers used by Indico. The dict specified here is passed to the `MULTIPASS_AUTH_PROVIDERS` setting of Flask-Multipass.

Default: {}

**IDENTITY_PROVIDERS**

A dict defining Flask-Multipass identity providers used by Indico to look up user information based on the data provided by an authentication provider. The dict specified here is passed to the `MULTIPASS_IDENTITY_PROVIDERS` setting of Flask-Multipass.

Default: {}

**PROVIDER_MAP**

If not specified, authentication and identity providers with the same name are linked automatically. The dict specified here is passed to the `MULTIPASS_PROVIDER_MAP` setting of Flask-Multipass.

Default: {}

**Cache**

**CACHE_BACKEND**

The backend used for caching. Valid backends are `redis`, `files`, and `memcached`.

To use the `redis` backend (recommended), you need to set `REDIS_CACHE_URL` to the URL of your Redis instance.

With the `files` backend, cache data is stored in `CACHE_DIR`, which always needs to be set, even when using a different cache backend since Indico needs to cache some data on disk.

To use the `memcached` backend, you need to install the `python-memcached` package from PyPI and set `MEMCACHED_SERVERS` to a list containing at least one memcached server.
Note: We only test Indico with the redis cache backend. While the other backends should work, we make no guarantees as they are not actively being used or tested.

Default: 'files'

**REDIS_CACHE_URL**
The URL of the redis server to use with the redis cache backend.

If the Redis server requires authentication, use a URL like this: `redis://unused:password@127.0.0.1:6379/1`

If no authentication is used (usually the case with a local Redis server), you can omit the user/password part: `redis://127.0.0.1:6379/1`

Default: None

**MEMCACHED_SERVERS**
The list of memcached servers (each entry is an ip:port string) to use with the memcached cache backend.

Default: []

**Celery**

**CELERY_BROKER**
The URL of the Celery broker (usually Redis of AMQP) used for communication between Indico and the Celery background workers.

We recommend using Redis as it is the easiest option, but you can check the Celery documentation on brokers for more information on the other possible brokers.

Default: None

**CELERY_RESULT_BACKEND**
The URL of the Celery result backend. If not set, the same backend as the broker is used. Indico currently does not use task results, and we recommend leaving this setting at its default.

Default: None

**CELERY_CONFIG**
A dict containing additional Celery settings.

```
Warning: This is an advanced setting that is rarely needed and we do not recommend using it unless you know exactly what you are doing! Changing Celery settings may break things or result in tasks not being executed without other changes (such as running additional celery workers on different queues).
```

One use case for this setting is routing certain tasks to a different queue, and then running multiple Celery workers for these queues.

```
CELERY_CONFIG = {
    'task_routes': {
        'indico_livesync.task.scheduled_update': {'queue': 'livesync'},
    }
}
```

Default: {}
SCHEDULED_TASK_OVERRIDE
A dict overriding the task schedule for specific tasks.

By default, all periodic tasks are enabled and use a schedule which we consider useful for most cases. Using this setting, you can override the default schedule.

The dict key is the name of the task and the value can be one of the following:

- None or False – disables the task completely
- A dictionary, as described in the Celery documentation on periodic tasks. The task should not be specified, as it is set automatically.
- A timedelta or crontab object which will just override the schedule without changing any other options of the task. Both classes are available in the config file by default.

**Note:** Use `indico celery inspect registered` to get a list of task names. Celery must be running for this command to work.

Default: `{}`

Customization

CUSTOMIZATION_DIR
The base path to the directory containing customizations for your Indico instance.

It is possible to override specific templates and add CSS and JavaScript for advanced customizations. When using this, be advised that depending on the modifications you perform things may break after an Indico update. Make sure to test all your modifications whenever you update Indico!

To include custom CSS and JavaScript, simply put `*.css` and `*.js` files into `<CUSTOMIZATION_DIR>/css`/`<CUSTOMIZATION_DIR>/js`. If there are multiple files, they will be included in alphabetical order, so prefixing them with a number (e.g. `00-base.css, 10-events.css`) is a good idea.

Static files may be added in `<CUSTOMIZATION_DIR>/files`. They can be referenced in templates through the `assets.custom` endpoint. In CSS/JS, the URL for them needs to be built manually (`/static/custom/files/...`).

For template customizations, see the description of CUSTOMIZATION_DEBUG as this setting is highly recommended to figure out where exactly to put customized templates.

Here is an example for a template customization that includes a custom asset and uses inheritance to avoid having to replace the whole template:

```latex
{% extends '~footer.html' %}

{% block footer_logo %}
  {%- set filename = 'cern_small_light.png' if dark else 'cern_small.png' -%}
  <a href="https://home.cern/" class="footer-logo">
    <img src="{{ url_for('assets.custom', filename=filename) }}" alt="CERN">
  </a>
{% endblock %}
```

Default: None

CUSTOMIZATION_DEBUG
Whether to log details for all customizable templates the first time they are accessed. The log message contains the path where you need to store the template; this path is relative to `<CUSTOMIZATION_DIR>/templates/`.

The log message also contains the full path of the original template in case you decide to copy it. However, instead of copying templates it is better to use Jinja inheritance where possible. To make this easier the log entry contains a “reference” path that can be used to reference the original template from the customized one.

**HELP_URL**

The URL used for the “Help” link in the footer.

Default: 'https://learn.getindico.io'

**LOGO_URL**

The URL to a custom logo. If unset, the default Indico logo is used.

Default: None

**CUSTOM_COUNTRIES**

A dict with country name overrides. This can be useful if the official ISO name of a country does not match what your Indico instance’s target audience expects for a country, e.g. due to political situations.

```
CUSTOM_COUNTRIES = {'KP': 'North Korea'}
```

Default: {}

---

**Database**

**SQLALCHEMY_DATABASE_URI**

The URI used to connect to the PostgreSQL database. For a local database, you can usually omit everything besides the database name: `postgresql:///indico`

If the database requires authentication and/or runs on a separate host, this form should be used: `postgresql://user:password@hostname/dbname`

**SQLALCHEMY_POOL_SIZE**

This setting configures SQLAlchemy’s connection pool. For details, check the Flask-SQLAlchemy documentation.

Default: 5

**SQLALCHEMY_POOL_RECYCLE**

This setting configures SQLAlchemy’s connection pool. For details, check the Flask-SQLAlchemy documentation.

Default: 120

**SQLALCHEMY_POOL_TIMEOUT**

This setting configures SQLAlchemy’s connection pool. For details, check the Flask-SQLAlchemy documentation.

Default: 10

---

**Development**

**Warning**: Do not turn on development settings in production. While we are not aware of serious security issues caused by these settings, they may slow down Indico or remove redundancies and thus make Indico not as stable as one would expect it to be in a production environment.
DEBUG
Enables debugging mode. If enabled, assets are not minified, error messages are more verbose and various other features are configured in a developer-friendly way.

**Do not enable debug mode in production.**

Default: False

DB_LOG
Enables real-time database query logging. When enabled, all database queries are sent to a socket where they can be read by the `db_log.py` script. To use the database logger, run `bin/utils/db_log.py` (only available when running Indico from a Git clone) in a separate terminal and all requests and verbose queries will be displayed there.

Default: False

PROFILE
Enables the Python profiler. The profiler output is stored in `<TEMP_DIR>/*.prof`.

Default: False

SMTP_USE_CELERY
If disabled, emails will be sent immediately instead of being handed to a Celery background worker. This is often more convenient during development as you do not need to run a Celery worker while still receiving emails sent from Indico. Disabling it may result in emails not being sent if the mail server is unavailable or some other failure happens during email sending. Because of this, the setting should never be disabled in a production environment.

Default: True

COMMUNITY_HUB_URL
The URL of the community hub. This should only be changed when using a local instance of Mereswine to debug the interface between Indico and Mereswine.

Default: 'https://hub.getindico.io'

DISABLE_CELERY_CHECK
Disables the warning about Celery not running or being outdated. When set to None, the warning is disabled when `DEBUG` is enabled; otherwise this setting enables/disables the warning regardless of debug mode.

Default: None

Directories

CACHE_DIR
The directory in which various data is cached temporarily. Must be accessible by the web server.

Default: '/opt/indico/cache'

LOG_DIR
The directory in which log files are stored. Can be overridden by using absolute paths in `logging.yaml`.

Default: '/opt/indico/log'

TEMP_DIR
The directory in which various temporary files are stored. Must be accessible by the web server.

Default: '/opt/indico/cache'
### Emails

**SMTP_SERVER**

The hostname and port of the SMTP server used for sending emails.

Default: ('localhost', 25)

**SMTP_LOGIN**

The username to send if the SMTP server requires authentication.

Default: None

**SMTP_PASSWORD**

The password to send if the SMTP server requires authentication.

Default: None

**SMTP_USE_TLS**

If enabled, STARTTLS will be used to use an encrypted SMTP connection.

Default: False

**SMTP_TIMEOUT**

The timeout in seconds after which a connection attempt to the SMTP server is aborted.

Default: 30

**NO_REPLY_EMAIL**

The email address used when sending emails to users to which they should not reply.

Default: None

**PUBLIC_SUPPORT_EMAIL**

The email address that is shown to users on the “Contact” page.

Default: None

**SUPPORT_EMAIL**

The email address of the technical manager of the Indico instance. Emails about unhandled errors/exceptions are sent to this address.

Default: None

### LaTeX

**XELATEX_PATH**

The full path to the xelatex program of TeXLive.

If it is installed in a directory in your $PATH, specifying its name without a path is sufficient.

If the path is not configured, any functionality that requires LaTeX on the server (such as generating the Book of Abstracts or exporting contributions to PDF) will be disabled.

Default: None

**STRICT_LATEX**

Enables strict mode for LaTeX rendering, in which case a non-zero status code is considered failure.

LaTeX is rather generous when it comes to using a non-zero exit code. For example, having an oversized image in an abstract is enough to cause one. It is generally not a good idea to enable strict mode as this will result in PDF generation to fail instead of creating a PDF that looks slightly uglier (e.g. a truncated image) than one that would succeed without a non-zero status code.

Default: False
Logging

**LOGGING_CONFIG_FILE**
The path to the logging config file. Unless an absolute path is specified, the path is relative to the location of the Indico config file after resolving symlinks.

Default: 'logging.yaml'

**SENTRY_DSN**
If you use Sentry for logging warnings/errors, you can specify the connection string here.

Default: None

**SENTRY_LOGGING_LEVEL**
The minimum level a log record needs to have to be sent to Sentry. If you do not care about warnings, set this to 'ERROR'.

Default: 'WARNING'

Security

**SECRET_KEY**
The secret key used to sign tokens in URLs. It must be kept secret under all circumstances.

When using Indico on a cluster of more than one worker, all machines need to have the same secret key.

The initial key is generated by the setup wizard, but if you have to regenerate it, the best way of doing so is running this snippet on a shell: `python -c 'import os; print repr(os.urandom(32))'`

Default: None

**SESSION_LIFETIME**
The duration of inactivity after which a session and its session cookie expires. If set to 0, the session cookie will be cleared when the browser is closed.

Default: 86400 * 31

Storage

**STORAGE_BACKENDS**
The list of backends that can be used to store/retrieve files.

Indico needs to store various files such as event attachments somewhere. By default only a filesystem based storage backend is available, but plugins could add additional backends. You can define multiple backends, but once a backend has been used, you MUST NOT remove it or all files stored in that backend will become unavailable.

To define a filesystem-based backend, use the string `fs:/base/path`. If you stopped using a backend, you can switch it to read-only mode by using `fs-readonly:` instead of `fs:`

Other backends may accept different options - see the documentation of these backends for details.

Default: [{'default': 'fs:/opt/indico/archive'}]

**ATTACHMENT_STORAGE**
The name of the storage backend used to store all kinds of attachments. Anything in this backend is write-once, i.e. once stored, files in it are never modified or deleted.

Changing this only affects new uploads; existing files are taken from the backend that was active when they were uploaded – which is also why you must not remove a backend from **STORAGE_BACKENDS** once it has been used.
Default: 'default'

**STATIC_SITE_STORAGE**
The name of the storage backend used to store “offline copies” of events. Files are written to this backend when generating an offline copy and deleted after a certain amount of time.

If not set, the `ATTACHMENT_STORAGE` backend is used.

Default: None

**System**

**BASE_URL**
This is the URL through which Indico is accessed by users. For production systems this should be an `https:/ /` URL and your web server should redirect all plain HTTP requests to HTTPS.

Default: None

**USE_PROXY**
This setting controls whether Indico runs behind a proxy or load balancer and should honor headers such as `X-Forwarded-For` to get the real IP address of the users accessing it.

The headers taken into account are:

- `X-Forwarded-For` – the IP address of the user
- `X-Forwarded-Proto` – the protocol used by the user
- `X-Forwarded-Host` – the hostname as specified in `BASE_URL` (can be omitted if the Host header is correct)

**Warning:** This setting **MUST NOT** be enabled if the server is accessible directly by untrusted clients without going through the proxy or users will be able to spoof their IP address by sending a custom `X-Forwarded-For` header. You need to configure your firewall so only requests coming from your proxy or load balancer are allowed.

Default: False

**ROUTE_OLD_URLS**
If you migrated from an older Indico version (v1.x), enable this option to redirect from the legacy URLs so external links keep working.

Default: False

**STATIC_FILE_METHOD**
This setting controls how static files (like attachments) are sent to clients.

Web servers are very good at doing this; much better and more efficient than Indico or the WSGI container, so this should be offloaded to your web server using this setting.

When using Apache with `mod_xsendfile` or lighttpd, set this to 'xsendfile' and of course enable xsendfile in your Apache config.

When using nginx, set this to (`'xaccelredirect', {'/opt/indico': '/.xsf/indico'}) and add an internal location handler to your nginx config to serve `/opt/indico` via `/opt/indico`:

```conf
location /.xsf/indico/ {
  internal;
  alias /opt/indico/;
}
```
The production installation instructions already configure this properly, so if you installed Indico using our guide, you only need to change this setting if you add e.g. a new storage backend in STORAGE_BACKENDS that stores the files outside /opt/indico.

Default: None

MAX_UPLOAD_FILE_SIZE
The maximum size of an uploaded file (in MB). A value of 0 disables the limit.
This limit is only enforced on the client side. For a hard limit that is enforced on the server, see MAX_UPLOAD_FILES_TOTAL_SIZE
Default: 0

MAXUPLOADFILES_TOTAL_SIZE
The maximum size (in MB) of all files uploaded in a single request (or to be more exact, any data contained in the body of a single request).
A value of 0 disables the limit, but most web servers also have limits which need to be configured as well (client_max_body_size in nginx) to allow very large uploads.
Default: 0

DEFAULT_LOCALE
The locale that is used by default for i18n. Valid values are en_GB, fr_FR, and es_ES.
Default: 'en_GB'

DEFAULT_TIMEZONE
The timezone that is used by default. Any timezone identifier such as Europe/Zurich or US/Central can be used.
Default: 'UTC'

ENABLE_ROOMBOOKING
Whether to enable the room booking system.
Default: False

PLUGINS
The list of Indico plugins to enable.
A list of all installed plugins can be displayed by the indico setup list-plugins command; see the guide linked above for details on how to enable plugins.
Default: set()

CATEGORY_CLEANUP
This setting specifies categories where events are automatically deleted a certain amount of days after they have been created.
For each entry, the key is the category id and the value the days after which an event is deleted.

Warning: This feature is mostly intended for “Sandbox” categories where users test Indico features. Since it is common for such categories to be used for real events nonetheless, we recommend enabling the “Event Header” in the category settings and clearly mention that the event will be deleted after a while.

Default: {}

WORKER_NAME
The name of the machine running Indico. The default value is usually fine unless your servers have ugly (e.g. auto-generated) hostnames and you prefer nicer names to show up in error emails.
FLOWER_URL
The URL of the Flower instance monitoring your Celery workers. If set, a link to it will be displayed in the admin area.

To use flower, install it using `pip install flower`, then start it using `indico celery flower`. By default it will listen on the same host as specified in `BASE_URL` (plain HTTP) on port 5555. Authentication is done using OAuth so only Indico administrators can access flower. You need to configure the allowed auth callback URLs in the admin area; otherwise authentication will fail with an OAuth error.

**Note:** The information displayed by Flower is usually not very useful. Unless you are very curious it is usually not worth using it.

Default: `None`

### 2.1.2 Authentication

Indico uses Flask-Multipass to handle authentication, searching for users in an external database, and externally managed groups. This means any Flask-Multipass authentication/identity provider can be used in Indico without any modifications to Indico itself.

For a description of the basic settings regarding local accounts (managed within Indico itself), see the general indico config documentation. This guide focuses solely on advanced authentication methods and how to configure them in Indico.

#### Configuration

**Authentication providers**

Authentication providers handle the login process, i.e. asking for user credentials or redirecting to an external site in case of SSO.

The `AUTH_PROVIDERS` setting is Indico’s equivalent to the `MULTIPASS_AUTH_PROVIDERS` setting of Flask-Multipass.

It must be set to a dict mapping a unique (internal) name of the auth provider (e.g. `mycompany-ldap`) to a dict of whatever data is needed for the given provider.

The following keys are available in the provider data:

- **type** – **Required.** The type of the provider. Valid values are e.g. `ldap`, `oauth`, `shibboleth`, and whatever custom providers you have installed.
- **title** – The title of the provider (shown on the login page). If omitted, the provider name is used.
- **default** – Must be set to `True` for exactly one form-based provider in case more than one such provider is used. The login form of the default provider is displayed when opening the login page so it should be the provider that most people use.
- Any provider-specific settings.
Identity providers

Identity providers get data about a user who logged in (based on the information passed on by the authentication provider) and also handle searching of external users and groups.

The `IDENTITY_PROVIDERS` setting is Indico’s equivalent to the `MULTIPASS_IDENTITY_PROVIDERS` setting of Flask-Multipass.

It must be set to a dict mapping a unique (internal) name of the identity provider (e.g. `mycompany-ldap`) to a dict of whatever data is needed for the given provider. Note that once an identity provider has been used, its name must not be changed.

The following keys are available in the provider data:

- **type** – **Required.** The type of the provider. Valid values are e.g. `ldap`, `oauth`, `shibboleth`, and whatever custom providers you have installed.
- **title** – The title of the provider (shown in the account list of the user profile). If omitted, the provider name is used.
- **default_group_provider** – If you have any providers which have group support (usually the case for LDAP), you should enable this for exactly one provider. This is used by legacy parts of Indico such as the room booking module which support groups but only take a group name and no information from which provider to get them.
- **trusted_email** – Set this to `True` if all email addresses received from the provider are trustworthy, i.e. if it is guaranteed that an email address actually belongs to the user (either because it’s coming from a trusted employee database or the provider is known to send verification emails). If an email is trusted, Indico will use it immediately to start the signup process or associate an existing account with a matching email address. Otherwise a verification email is sent to prove that the user has access to the email address, which is less user-friendly but extremely important to prevent malicious takeovers of Indico accounts.
- **moderated** – Set this to `True` if you want to require manual approval of the registration by an Indico admin. This results in the same workflow as `LOCAL_MODERATION` in case of local accounts.
- **synced_fields** – This may be set in no more than once identity provider and enables user data synchronization. Its value should be a set of user attributes that can be synchronized during login. Indico does not support synchronizing email addresses; only the following attributes can be synchronized: `first_name`, `last_name`, `affiliation`, `phone`, `address`
- **mapping** – A dictionary that maps between keys given by the identity provider and keys expected by Indico for user information. The key of each entry is the Indico-side attribute name; the value is the key under which the data is exposed by the provider. Indico can take user information from the following keys: `first_name`, `last_name`, `email`, `affiliation`, `phone`, `address`. For example, this mapping would use the `givenName` provided by the identity provider to populate the user’s `first_name` in Indico:

```
'mapping': {'first_name': 'givenName'}
```

- **identity_info_keys** – By default, all six attributes listed above will be used if the provider has them (either directly or in some other field specified in the `mapping`). If you want to restrict the data from a provider (e.g. because the value it provides is known to be useless/incorrect), you can set this to a set containing only the attributes you want to use. Note that external user search requires email addresses, so if you exclude email addresses here, users from this provider will never appear in search results.
- Any provider-specific settings.
Links between providers

By default, authentication and identity providers with the same name are linked together. If this is not what you want, you can use the `PROVIDER_MAP` setting to manually link providers. This is useful for advanced cases where you have e.g. both a login form to enter LDAP credentials and a SSO provider, but want to have a single LDAP identity provider that can use the username from either SSO or the LDAP login. In this case you would link both authentication providers to the same identity provider.

Specific providers

LDAP

The `ldap` authentication/identity providers are available by default, but to use them you need to install the `python-ldap` library using `pip install python-ldap`.

**Note:** `python-ldap` has some extra system dependencies (openldap and libsasl). How to install them (apt, yum, etc.) depends on your Linux distribution. The package names are usually `libsasl2-dev` or `libsasl-dev` and `openldap-dev` (or `-devel` on some distros). If one of these libraries is missing, `pip` will fail when installing `python-ldap`. Simply re-run the command after installing the missing library.

Once everything is installed, you can add the LDAP-related settings to your `indico.conf`. Below is an example based on the LDAP config we use at CERN with Active Directory; you can copy this as a starting point for your own config and then adapt it to your own environment:

```python
_ldap_config = {
    'uri': 'ldaps://...',
    'bind_dn': 'cn=***,OU=Users,OU=Organic Units,DC=cern,DC=ch',
    'bind_password': '***',
    'timeout': 30,
    'verify_cert': True,
    'page_size': 1500,
    'uid': 'cn',
    'user_base': 'DC=cern,DC=ch',
    'user_filter': '(objectCategory=user)',
    'gid': 'cn',
    'group_base': 'OU=Workgroups, DC=cern, DC=ch',
    'group_filter': '(objectCategory=group)',
    'member_of_attr': 'memberOf',
    'ad_group_style': True
}

AUTH_PROVIDERS = {
    'ldap': {
        'type': 'ldap',
        'title': 'LDAP',
        'ldap': _ldap_config,
        'default': True
    }
}
```

(continues on next page)
IDENTITY_PROVIDERS = {
    'ldap': {
        'type': 'ldap',
        'title': 'LDAP',
        'ldap': _ldap_config,
        'mapping': {
            'first_name': 'givenName',
            'last_name': 'sn',
            'email': 'mail',
            'affiliation': 'company',
            'phone': 'telephoneNumber'
        },
        'trusted_email': True,
        'default_group_provider': True,
        'synced_fields': {'first_name', 'last_name', 'affiliation', 'phone', 'address'}
    }
}

The LDAP-specific config uses the following keys:

- **uri** – **Required.** The URI referring to the LDAP server including the protocol and the port. Use ldaps:// for LDAP over SSL/TLS and ldap:// with the starttls option for a plain LDAP connection with TLS negotiation. The port can be omitted if the LDAP server listens on the default port (636 for LDAP over SSL and 389 for a plain LDAP connection with TLS negotiation).

- **bind_dn** – **Required.** The distinguished name to bind to the LDAP directory.

- **bind_password** – **Required.** The password to use together with the bind_dn to login to the LDAP server.

- **timeout** – The delay in seconds to wait for a reply from the LDAP server (set to -1 to disable). Default: 30

- **verify_cert** – Whether to verify the TLS certificate of the LDAP server. Default: True

- **starttls** – Whether to use STARTTLS to switch to an encrypted connection. Ignored with an ldaps:// URI. Default: False

- **page_size** – The limit of entries to retrieve at once for a search. 0 means no size limit. It is recommended to have at most the size limit imposed by the server. Default: 1000

- **uid** – The attribute whose value is used as an identifier for the user (typically the username). This attribute must be a single-valued attribute whose value is unique for each user. If the attribute is multi-valued, only the first one retrieved will be returned. Default: 'uid'

- **user_base** – **Required.** The base node for all the nodes which might contain a user.

- **user_filter** – A valid LDAP filter which will select exclusively all users in the subtree from the user_base. The combination of the user_base and the user_filter must match exclusively all the users. Default: '(objectClass=person)'

- **gid** – The attribute whose value is used as an identifier for the group (typically the group’s name). This attribute must be a single-valued attribute whose value is unique for each group. If the attribute is multi-valued, only the first one retrieved will be returned. Default: 'cn'

- **group_base** – **Required.** The base node for all the nodes which might contain a group.

- **group_filter** – A valid LDAP filter which will select exclusively all groups in the subtree from the group_base. The combination of the group_base and the group_filter must match exclusively all the groups. Default: '(objectClass=groupOfNames)'
• **member_of_attr** – The multi-valued attribute of a user containing the list of groups the user is a member of. Default: 'memberOf'

**Note:** In case of SLAPD/OpenLDAP, the **member of** attribute must be enabled. While it is not enabled by default, the majority of servers will have it enabled. A simple `ldapsearch` for a user member of any group should show if that is the case. If not, you can check this article on information how to enable it on your LDAP server. Note that unless you manage the LDAP server, you need to ask the administrator of that server to do that.

• **ad_group_style** – Whether the server uses Active-Directory-style groups or not. This is only used when checking if a user is a member of a group. If enabled, the code will take advantage of the `tokenGroups` attribute of a user to check for nested group membership. Otherwise, it will only look through the values of the `member_of_attr`, which should also work for Active Directory, but only for direct membership. Default: False

**SAML / Shibboleth**

The **shibboleth** authentication/identity providers are available by default, but due to how the protocol works you need to use the Apache webserver to use SAML authentication provider.

You can find guides on how to set it up for **CentOS** and **Debian**.

If you also have an LDAP server, it may be a good idea to use the **shibboleth** authentication provider and connect it to an **ldap** identity provider. This way the user information is retrieved from LDAP based on a unique identifier of the user that comes from SAML, and you can still use the search and group functionality provided by LDAP.
CHAPTER 3

Plugins

Indico can be extended through plugins, standalone packages of code that do not require any modifications to the Indico core itself. A plugin can perform something very simple such as adding a new command to the Indico CLI to more complex functionalities like introducing new payment methods, chat integration, etc. We suggest that you first have a look at Getting started and then head over to the more advance topics in the table of contents.

3.1 Extending Indico with plugins

Indico can be extended through plugins, standalone packages of code that do not require any modifications to the Indico core itself. A plugin can perform something very simple such as adding a new command to the Indico CLI to more complex functionalities like introducing new payment methods, chat integration, etc. We suggest that you first have a look at Getting started and then head over to the more advance topics in the table of contents.

3.1.1 Getting started with Indico plugins

Todo: Write a REAL, simple example of a plugin. Include link to Github repo.

Example plugin

The following is a minimal plugin that makes use of all capababilites of the plugin API. The display name of the plugin is defined by the first line of the docstring and the description by the rest of it. The plugin may override signal handlers to hook into Indico and aditionally run any initialization needed. For example, it will add some command to Indico CLI, extend the shell context and register some assets. Also, init is used to inject CSS and JS bundles outside of the plugin scope.

```python
class ExamplePlugin(IndicoPlugin):
    # Example Plugin
```

(continues on next page)
An example plugin that demonstrates the capabilities of the new Indico plugin system.

```python
settings_form = SettingsForm

def init(self):
    super(ExamplePlugin, self).init()
    self.inject_bundle('main.js')

def get_blueprints(self):
    return blueprint

def add_cli_command(self, manager):
    @manager.command
    @with_plugin_context(self)
    def example():
        """Example command from example plugin""
        print 'example plugin says hi', current_plugin
        if self.settings.get('show_message'):
            print self.settings.get('dummy_message')

    def extend_shell_context(self, add_to_context):
        add_to_context('bar', name='foo', doc='foobar from example plugin', color='magenta!')
```

The plugin can specify its settings via a `IndicoForm`:

```python
class SettingsForm(IndicoForm):
    dummy_message = StringField('Dummy Message')
    show_message = BooleanField('Show Message')
```

The plugin can also specify request handlers and templates. Templates will be loaded from a `templates` folder within your plugin folder. Your plugin can even load templates from other modules by prefixing the name of the template `other_plugin:example` with `render_template()`.

```python
class WPExample(WPDecorated):
    def _get_body(self, params):
        return render_plugin_template('example.html', **params)

class RHExample(RH):
    def _process(self):
        return WPExample(self, foo='bar').display()

class RHTest(RH):
    def _process(self):
        return render_plugin_template('test.html')
```

```python
blueprint = IndicoPluginBlueprint('example', __name__)
blueprint.add_url_rule('/example', 'example', view_func=RHExample)
blueprint.add_url_rule('/example/x', 'example', view_func=RHExample)
blueprint.add_url_rule('/test', 'test', view_func=RHTest)
```
3.1.2 Plugin API reference

Indico’s plugin system allows you to extend indico with additional modules which can be installed separately and do not require any modifications to the indico core itself.

```python
class indico.core.plugins.IndicoPlugin(plugin_engine, app)
Bases: flask_pluginengine.plugin.Plugin

Base class for an Indico plugin

All your plugins need to inherit from this class. It extends the Plugin class from Flask-PluginEngine with useful indico-specific functionality that makes it easier to write custom plugins.

When creating your plugin, the class-level docstring is used to generate the friendly name and description of a plugin. Its first line becomes the name while everything else goes into the description.

This class provides methods for some of the more common hooks Indico provides. Additional signals are defined in signals and can be connected to custom functions using connect().

```acl_event_settings = frozenset([])``
A set containing the names of event-specific settings which store ACLs

```acl_settings = frozenset([])``
A set containing the names of settings which store ACLs

```category = None``
The group category that the plugin belongs to

```configurable = False``
If the plugin should link to a details/config page in the admin interface

```default_event_settings = {}``
A dictionary containing default values for event-specific settings

```default_settings = {}``
A dictionary containing default values for settings

```default_user_settings = {}``
A dictionary containing default values for user-specific settings

```event_settings``
classmethod(function) -> method

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```
class C: @classmethod def f(cls, arg1, arg2, ...):
  ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

```event_settings_converters = {}``
A dict containing custom converters for event-specific settings

```get_blueprints()``
 Return blueprints to be registered on the application

3.1. Extending Indico with plugins
A single blueprint can be returned directly, for multiple blueprint you need to yield them or return an iterable.

**get_vars_js()**
Return a dictionary with variables to be added to vars.js file

**init()**
Called when the plugin is being loaded/initialized.

If you want to run custom initialization code, this is the method to override. Make sure to call the base method or the other overridable methods in this class will not be called anymore.

**inject_bundle**(name, view_class=None, subclasses=True, condition=None)
Injects an asset bundle into Indico’s pages

**Parameters**
- **name** – Name of the bundle
- **view_class** – If a WP class is specified, only inject it into pages using that class
- **subclasses** – also inject into subclasses of **view_class**
- **condition** – a callable to determine whether to inject or not. only called, when the **view_class** criterion matches

**inject_vars_js()**
Returns a string that will define variables for the plugin in the vars.js file

**settings**
classmethod(function) -> method
Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```python
class C: @classmethod def f(cls, arg1, arg2, ...):
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

**settings_converters = {}**
A dict containing custom converters for settings

**settings_form = None**
WTForm for the plugin’s settings (requires configurable=True). All fields must return JSON-serializable types.

**settings_form_field_opts = {}**
A dictionary which can contain the kwargs for a specific field in the **settings_form**.

**strict_settings = True**
If **settings**, **event_settings** and **user_settings** should use strict mode, i.e. only allow keys in **default_settings**, **default_event_settings** or **default_user_settings** (or the related acl_settings sets). This should not be disabled in most cases; if you need to store arbitrary keys, consider storing a dict inside a single top-level setting.
**template_hook** *(name, receiver, priority=50, markup=True)*

Registers a function to be called when a template hook is invoked.

For details see [register_template_hook()](#).

**translation_domain**

Return the domain for this plugin’s translation_path.

**translation_path**

Return translation files to be used by the plugin. By default, get `<root_path>/translations`, unless it does not exist.

**user_settings**

```
classmethod(function) -> method
```

Convert a function to be a class method.

A class method receives the class as implicit first argument, just like an instance method receives the instance. To declare a class method, use this idiom:

```python
class C:  @classmethod def f(cls, arg1, arg2, ...):
...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class. If a class method is called for a derived class, the derived class object is passed as the implied first argument.

Class methods are different than C++ or Java static methods. If you want those, see the staticmethod builtin.

**user_settings_converters** = {}

A dict containing custom converters for user-specific settings.

### class **indico.core.plugins.IndicoPluginBlueprint** *(name, *args, **kwargs)*

Bases: `flask_pluginengine.mixins.PluginBlueprintMixin`, `indico.web.flask.wrappers.IndicoBlueprint`

The Blueprint class all plugins need to use.

It contains the necessary logic to run the blueprint’s view functions inside the correct plugin context and to make the static folder work.

**make_setup_state** *(app, options, first_registration=False)*

Creates an instance of `BlueprintSetupState()` object that is later passed to the register callback functions. Subclasses can override this to return a subclass of the setup state.

### class **indico.core.plugins.IndicoPluginBlueprintSetupState** *(blueprint, app, options, first_registration)*

Bases: `flask_pluginengine.mixins.PluginBlueprintSetupStateMixin`, `indico.web.flask.wrappers.IndicoBlueprintSetupState`

**add_url_rule** *(rule, endpoint=None, view_func=None, **options)*

A helper method to register a rule (and optionally a view function) to the application. The endpoint is automatically prefixed with the blueprint’s name.

### class **indico.core.plugins.PluginCategory**

Bases: `unicode`, `indico.util.struct.enum.IndicoEnum`

**indico.core.plugins.get_plugin_template_module** *(template_name, **context)*

Like `get_template_module()`, but using plugin templates

**indico.core.plugins.plugin_url_rule_to_js** *(endpoint)*

Like `url_rule_to_js()` but prepending plugin name prefix to the endpoint

### 3.1. Extending Indico with plugins
indico.core.plugins.url_for_plugin(endpoint, *targets, **values)

Like url_for() but prepending 'plugin_' to the blueprint name.

### 3.1.3 Hooking into Indico using Signals

**Contents**

- Hooking into Indico using Signals
  - indico.core.signals
    - indico.core.signals.acl
    - indico.core.signals.agreements
    - indico.core.signals.attachments
    - indico.core.signals.category
    - indico.core.signals.event
    - indico.core.signals.event_management
    - indico.core.signals.menu
    - indico.core.signals.plugin
    - indico.core.signals.rb
    - indico.core.signals.rh
    - indico.core.signals.users

Signals allow you to hook into certain parts of Indico without adding any code to the core (which is something a plugin can and should not do). Each signal has a *sender* which can be any object (depending on the signal) and possibly some keyword arguments. Some signals also make use of their return value or even require one. Check the documentation of each signal on how it’s used.

To avoid breakage with newer versions of Indico, it is highly advised to always accept extra **kwargs in your signal receiver. For example, a receiver function could look like this:

```python
def receiver(sender, something, **kwargs):
    do_stuff_with(something)
```

**indico.core.signals**

**indico.core.signals.add_form_fields**

Lets you add extra fields to a form. The *sender* is the form class and should always be specified when subscribing to this signal.

The signal handler should return one or more 'name', Field tuples. Each field will be added to the form as ext__<name> and is automatically excluded from the form’s data property and its populate_obj method.

To actually process the data, you can use e.g. the form_validated signal and then store it in flask.g until another signal informs you that the operation the user was performing has been successful.

**indico.core.signals.after_commit**

Called after an SQL transaction has been committed. Note that the session is in ‘committed’ state when this signal is called, so no SQL can be emitted while this signal is being handled.
indico.core.signals.**after_process**
Called after an Indico request has been processed. This signal is executed for both RH classes and legacy JSON-RPC services.

indico.core.signals.**app_created**
Called when the app has been created. The *sender* is the flask app.

indico.core.signals.**db_schema_created**
Executed when a new database schema is created. The *sender* is the name of the schema.

indico.core.signals.**form_validated**
Triggered when an IndicoForm was validated successfully. The *sender* is the form object.

This signal may return `False` to mark the form as invalid even though WTForms validation was successful. In this case it is highly recommended to mark a field as erroneous or indicate the error in some other way.

indico.core.signals.**get_conditions**
Expected to return one or more classes inheriting from *Condition*. The *sender* is a string (or some other object) identifying the context. The additional kwargs passed to this signal depend on the context.

indico.core.signals.**get_fields**
Expected to return *BaseField* subclasses. The *sender* is an object (or just a string) identifying for what to get fields. This signal should never be registered without restricting the sender to ensure only the correct field types are returned.

indico.core.signals.**get_placeholders**
Expected to return one or more *Placeholder* objects. The *sender* is a string (or some other object) identifying the context. The additional kwargs passed to this signal depend on the context.

indico.core.signals.**get_storage_backends**
Expected to return one or more Storage subclasses.

indico.core.signals.**import_tasks**
Called when Celery needs to import all tasks. Use this signal if you have modules containing task registered using one of the Celery decorators but don’t import them anywhere. The signal handler should only import these modules and do nothing else.

indico.core.signals.acl

indico.core.signals.acl.**can_access**
Called when *ProtectionMixin.can_access* is used to determine if a user can access something or not.

The *sender* is the type of the object that’s using the mixin. The actual instance is passed as *obj*. The *user* and *allow_admin* arguments of *can_access* are passed as kwargs with the same name.

The *authorized* argument is `None` when this signal is called at the beginning of the access check and `True` or `False` at the end when regular access rights have already been checked. For expensive checks (such as anything involving database queries) it is recommended to skip the check while *authorized* is `None` since the regular access check is likely to be cheaper (due to ACLs being preloaded etc).

If the signal returns `True` or `False`, the access check succeeds or fails immediately. If multiple subscribers to the signal return contradictory results, `False` wins and access is denied.

indico.core.signals.acl.**can_manage**
Called when *ProtectionMixin.can_manage* is used to determine if a user can manage something or not.

The *sender* is the type of the object that’s using the mixin. The actual instance is passed as *obj*. The *user*, *permission*, *allow_admin*, *check_parent* and *explicit_permission* arguments of *can_manage* are passed as kwargs with the same name.

---

3.1. Extending Indico with plugins
If the signal returns `True` or `False`, the access check succeeds or fails without any further checks. If multiple subscribers to the signal return contradictory results, `False` wins and access is denied.

**indico.core.signals.acl.entry_changed**

Called when an ACL entry is changed.

The `sender` is the type of the object that’s using the mixin. The actual instance is passed as `obj`. The `User`, `GroupProxy` or `EmailPrincipal` is passed as `principal` and `entry` contains the actual ACL entry (a `PrincipalMixin` instance) or `None` in case the entry was deleted. `is_new` is a boolean indicating whether the given principal was in the ACL before. If `quiet` is `True`, signal handlers should not perform noisy actions such as logging or sending emails related to the change.

If the ACL uses permissions, `old_data` will contain a dictionary of the previous permissions (see `PrincipalPermissionsMixin.current_data`).

**indico.core.signals.acl.get_management_permissions**

Expected to return `ManagementPermission` subclasses. The `sender` is the type of the object the permissions may be used for. Functions subscribing to this signal **MUST** check the sender by specifying it using the first argument of `connect_via()` or by comparing it inside the function.

**indico.core.signals.acl.protection_changed**

Called when the protection mode of an object is changed.

The `sender` is the type of the object that’s using the mixin. The actual instance is passed as `obj`. The old protection mode is passed as `old_mode`, the new mode as `mode`.

**indico.core.signals.agreements**

**indico.core.signals.agreements.get_definitions**

Expected to return a list of `AgreementDefinition` classes.

**indico.core.signals.attachments**

**indico.core.signals.attachments.attachment_accessed**

Called when an attachment is accessed. The `sender` is the `Attachment` that was accessed. The user who accessed the attachment is passed in the `user` kwarg. The `from_preview` kwarg will be set to `True` if the download link on the preview page was used to access the attachment or if the attachment was loaded to be displayed on the preview page (opening the preview itself already sends this signal with `from_preview=False`).

**indico.core.signals.attachments.attachment_created**

Called when a new attachment is created. The `sender` object is the new `Attachment`. The user who created the attachment is passed in the `user` kwarg.

**indico.core.signals.attachments.attachment_deleted**

Called when an attachment is deleted. The `sender` object is the `Attachment` that was deleted. The user who deleted the attachment is passed in the `user` kwarg.

**indico.core.signals.attachments.attachment_updated**

Called when an attachment is updated. The `sender` is the `Attachment` that was updated. The user who updated the attachment is passed in the `user` kwarg.

**indico.core.signals.attachments.folder_created**

Called when a new attachment folder is created. The `sender` is the new `AttachmentFolder` object. The user who created the folder is passed in the `user` kwarg. This signal is never triggered for the internal default folder.

**indico.core.signals.attachments.folder_deleted**

Called when a folder is deleted. The `sender` is the `AttachmentFolder` that was deleted. The user who deleted the folder is passed in the `user` kwarg.
indico.core.signals.attachments.folder_updated
Called when a folder is updated. The sender is the AttachmentFolder that was updated. The user who updated the folder is passed in the user kwarg.

indico.core.signals.attachments.get_file_previewers
Expected to return one or more Previewer subclasses.

**indico.core.signals.category**

indico.core.signals.category.created
Called when a new category is created. The sender is the new category.

indico.core.signals.category.deleted
Called when a category is deleted. The sender is the category.

indico.core.signals.category.moved
Called when a category is moved into another category. The sender is the category and the old parent category is passed in the old_parent kwarg.

indico.core.signals.category.updated
Called when a new category is updated. The sender is the new category.

**indico.core.signals.event**

indico.core.signals.event.abstract_created
Called when a new abstract is created. The sender is the new abstract.

indico.core.signals.event.abstract_deleted
Called when an abstract is deleted. The sender is the abstract.

indico.core.signals.event.abstract_state_changed
Called when an abstract is withdrawn. The sender is the abstract.

indico.core.signals.event.abstract_updated
Called when an abstract is modified. The sender is the abstract.

indico.core.signals.event.cloned
Called when an event is cloned. The sender is the Event object of the old event, the new event is passed in the new_event kwarg.

indico.core.signals.event.contribution_created
Called when a new contribution is created. The sender is the new contribution.

indico.core.signals.event.contribution_deleted
Called when a contribution is deleted. The sender is the contribution.

indico.core.signals.event.contribution_updated
Called when a contribution is modified. The sender is the contribution. A dict containing old, new tuples for all changed values is passed in the changes kwarg.

indico.core.signals.event.created
Called when a new event is created. The sender is the new Event.

indico.core.signals.event.deleted
Called when an event is deleted. The sender is the event object. The user kwarg contains the user performing the deletion if available.

3.1. Extending Indico with plugins 69
indico.core.signals.event.generate_ticket_qr_code
Called when generating the QR code for a ticket. The data included in the QR code is passed in the ticket_data kwarg and may be modified.

indico.core.signals.event.get_feature_definitions
Expected to return EventFeature subclasses.

indico.core.signals.event.get_log_renderers
Expected to return EventLogRenderer classes.

indico.core.signals.event.is_ticket_blocked
Called when resolving whether Indico should let a registrant download their ticket. The sender is the registrant’s Registration object.

If this signal returns True, the user will not be able to download their ticket. Any badge containing a ticket-specific placeholder such as the ticket qr code is considered a ticket, and the restriction applies to both users trying to get their own ticket and managers trying to get a ticket for a registrant.

indico.core.signals.event.is_ticketing_handled
Called when resolving whether Indico should send tickets with e-mails or it will be handled by other module. The sender is the RegistrationForm object.

If this signal returns True, no ticket will be emailed on registration.

indico.core.signals.event.metadata_postprocess
Called right after a dict-like representation of an event is created, so that plugins can add their own fields.

The sender is a string parameter specifying the source of the metadata. The event kwarg contains the event object. The metadata is passed in the data kwarg.

The signal should return a dict that will be used to update the original representation (fields to add or override).

indico.core.signals.event.moved
Called when an event is moved to a different category. The sender is the event, the old category is in the old_parent kwarg.

indico.core.signals.event.note_added
Called when a note is added. The sender is the note.

indico.core.signals.event.note_deleted
Called when a note is deleted. The sender is the note.

indico.core.signals.event.note_modified
Called when a note is modified. The sender is the note.

indico.core.signals.event.person_updated
Called when an EventPerson is modified. The sender is the EventPerson.

indico.core.signals.event.print_badge_template
Called when printing a badge template. The registration form is passed in the regform kwarg, the list of registration objects are passed in the registrations kwarg.

indico.core.signals.event.registration_checkin_updated
Called when the checkin state of a registration changes. The sender is the Registration object.

indico.core.signals.event.registration_created
Called when a new registration has been created. The sender is the Registration object. The management kwarg is set to True if the registration was created from the event management area.

indico.core.signals.event.registration_deleted
Called when a registration is removed. The sender is the Registration object.
indico.core.signals.event\_registration\_form\_created
Called when a new registration form is created. The sender is the Registration\_Form object.

indico.core.signals.event\_registration\_form\_deleted
Called when a registration form is removed. The sender is the Registration\_Form object.

indico.core.signals.event\_registration\_personal\_data\_modified
Called when the registration personal data is modified. The sender is the Registration object; the change is passed in the change kwarg.

indico.core.signals.event\_registration\_state\_updated
Called when the state of a registration changes. The sender is the Registration object; the previous state is passed in the previous\_state kwarg.

indico.core.signals.event\_registration\_updated
Called when a registration has been updated. The sender is the Registration object. The management kwarg is set to True if the registration was updated from the event management area.

indico.core.signals.event\_session\_block\_deleted
Called when a session block is deleted. The sender is the session block. This signal is called before the db. session.delete() on the block is executed.

indico.core.signals.event\_session\_deleted
Called when a session is deleted. The sender is the session.

indico.core.signals.event\_session\_updated
Called when a session is updated. The sender is the session.

indico.core.signals.event\_sidemenu
Expected to return Menu\_Entry\_Data objects to be added to the event side menu. A single entry can be returned directly, multiple entries must be yielded.

indico.core.signals.event\_subcontribution\_created
Called when a new subcontribution is created. The sender is the new subcontribution.

indico.core.signals.event\_subcontribution\_deleted
Called when a subcontribution is deleted. The sender is the subcontribution.

indico.core.signals.event\_subcontribution\_updated
Called when a subcontribution is modified. The sender is the subcontribution.

indico.core.signals.event\_times\_changed
Called when the times of a scheduled object (contribution, break or session block) change, either by a change in duration or start time. The sender is the type of the object; the timetable entry is passed as entry and the object is passed as obj. Information about the changes are passed as changes which is a dict containing old/new tuples for start\_dt, duration and end\_dt. If an attribute did not change, it is not included in the dict. If the time of the event itself changes, entry is None and obj contains the Event.

indico.core.signals.event\_timetable\_buttons
Expected to return a list of tuples (‘button\_name’, ‘js\_call\_class’). Called when building the timetable view.

indico.core.signals.event\_timetable\_entry\_created
Called when a new timetable entry is created. The sender is the new entry.

indico.core.signals.event\_timetable\_entry\_deleted
Called when a timetable entry is deleted. The sender is the entry. This signal is triggered right before the entry deletion is performed.

indico.core.signals.event\_timetable\_entry\_updated
Called when a timetable entry is updated. The sender is the entry. A dict containing old, new tuples for all changed values is passed in the changes kwarg.

3.1. Extending Indico with plugins
**indico.core.signals.event**.type_changed
Called when the type of an event is changed. The sender is the event, the old type is passed in the old_type kwarg.

**indico.core.signals.event**.updated
Called when basic data of an event is updated. The sender is the event. A dict of changes is passed in the changes kwarg, with (old, new) tuples for each change. Note that the person_links change may happen with old and new being the same lists for technical reasons. If the key is present, it should be assumed that something changed (usually the order or some data on the person link).

**indico.core.signals.event_management**

**indico.core.signals.event_management**.get_cloners
Expected to return one or more EventCloner subclasses implementing a cloning operation for something within an event.

**indico.core.signals.event_management**.image_created
Called when a new image is created. The sender object is the new ImageFile. The user who uploaded the image is passed in the user kwarg.

**indico.core.signals.event_management**.image_deleted
Called when an image is deleted. The sender object is the ImageFile that is about to be deleted. The user who uploaded the image is passed in the user kwarg.

**indico.core.signals.event_management**.management_url
Expected to return a URL for the event management page of the plugin. This is used when someone who does not have event management access wants to go to the event management area. He is then redirected to one of the URLs returned by plugins, i.e. it is not guaranteed that the user ends up on a specific plugin’s management page. The signal should return None if the current user (available via session.user) cannot access the management area. The sender is the event object.

**indico.core.signals.menu**

**indico.core.signals.menu**.items
Expected to return one or more SideMenuItem to be added to the side menu. The sender is an id string identifying the target menu.

**indico.core.signals.menu**.sections
Expected to return one or more SideMenuSection objects to be added to the side menu. The sender is an id string identifying the target menu.

**indico.core.signals.plugin**

**indico.core.signals.plugin**.cli
Expected to return one or more click commands/groups. If they use indico.cli.core.cli_command / indico.cli.core.cli_group they will be automatically executed within a plugin context and run within a Flask app context by default.

**indico.core.signals.plugin**.get_blueprints
Expected to return one or more IndicoPluginBlueprint-based blueprints which will be registered on the application. The Blueprint must be named either PLUGINNAME or compat_PLUGINNAME.

**indico.core.signals.plugin**.get_conference_themes
Expected to return (name, css, title) tuples for conference stylesheets. name is the internal name used
for the stylesheet which will be stored when the theme is selected in an event. css is the location of the CSS file, relative to the plugin's static folder. title is the title displayed to the user when selecting the theme.

indico.core.signals.plugin.get_event_request_definitions
Expected to return one or more RequestDefinition subclasses.

indico.core.signals.plugin.get_event_themes_files
Expected to return the path of a themes yaml containing event theme definitions.

indico.core.signals.plugin.get_template_customization_paths
Expected to return the absolute path to a directory containing template overrides. This signal is called once during initialization so it should not use any data that may change at runtime. The behavior of a customization path returned by this function is exactly like <CUSTOMIZATION_DIR>/templates, but it has lower priority than the one from the global customization dir.

indico.core.signals.plugin.inject_bundle
Expected to return a list of bundle names which are loaded after all the rest. The sender is the WP class of the page.

indico.core.signals.plugin.schema_post_dump
Called when a marshmallow schema is dumped. The sender is the schema class and code using this signal should always specify it. The signal is called with the following arguments:

- many – bool indicating whether the data was dumped with many=True or not
- data – the dumped data. this is guaranteed to be a list; in case of many=False it is guaranteed to contain exactly one element
- orig – the original data before dumping. just like data it is always a list

If a plugin wants to modify the data returned when dumping, it may do so by modifying the contents of data.

indico.core.signals.plugin.schema_post_load
Called after a marshmallow schema is loaded. The sender is the schema class and code using this signal should always specify it. The signal is called with the following arguments:

- data – the data returned by marshmallow; this is usually a dict which may contain more complex data types than those valid in JSON

If a plugin wants to modify the resulting data, it may do so by modifying the contents of data.

indico.core.signals.plugin.schema_pre_load
Called when a marshmallow schema is loaded. The sender is the schema class and code using this signal should always specify it. The signal is called with the following arguments:

- data – the raw data passed to marshmallow; this is usually a dict of raw json/form data coming from the user, so it can have all types valid in JSON

If a plugin wants to modify the data the schema will eventually load, it may do so by modifying the contents of data.

indico.core.signals.plugin.shell_context
Called after adding stuff to the indico shell context. Receives the add_to_context and add_to_context_multi keyword args with functions which allow you to add custom items to the context.

indico.core.signals.plugin.template_hook
Expected to return a (is_markup, priority, value) tuple. The returned value will be inserted at the location where this signal is triggered; if multiple receivers are connected to the signal, they will be ordered by priority. If is_markup is True, the value will be wrapped in a Markup object which will cause it to be rendered as HTML. The sender is the name of the actual hook. The keyword arguments depend on the hook.
### Indico Documentation, Release 2.3-dev

**indico.core.signals.rb**

- **booking_created**
  Executed after a booking has been successfully created. The *sender* is the new *Reservation* object.

- **booking_deleted**
  Executed after a booking has been deleted. The *sender* is the *Reservation* object.

- **booking_occurrence_state_changed**
  Executed after the state of a booking occurrence changed. The *sender* is the *ReservationOccurrence* object.

- **booking_state_changed**
  Executed after a booking has been cancelled/rejected/accepted. The *sender* is the *Reservation* object.

**indico.core.signals.rh**

- **before_process**
  Executed right before _process of an RH instance is called. The *sender* is the RH class, the current instance is passed in *rh*. If a signal handler returns a value, the original _process method will not be executed. If multiple signal handlers return a value, an exception is raised.

- **check_access**
  Executed right after _check_access of an RH instance has been called unless the access check raised an exception. The *sender* is the RH class, the current instance is passed in *rh*.

- **process**
  Executed right after _process of an RH instance has been called. The *sender* is the RH class, the current instance is passed in *rh*. The return value of _process is available in *result* and if a signal handler returns a value, it will replace the original return value. If multiple signals handlers return a value, an exception is raised.

- **process_args**
  Executed right after _process_args of an RH instance has been called. The *sender* is the RH class, the current instance is passed in *rh*. The return value of _process_args is available in *result*.

**indico.core.signals.users**

- **email_added**
  Called when a new email address is added to a user. The *sender* is the user object and the email address is passed in the *email* kwarg.

- **merged**
  Called when two users are merged. The *sender* is the main user while the merged user (i.e. the one being deleted in the merge) is passed via the *source* kwarg.

- **preferences**
  Expected to return a *ExtraUserPreferences* subclass which implements extra preferences for the user preference page. The *sender* is the user for whom the preferences page is being shown which might not be the currently logged-in user!

- **registered**
  Called once a user registers (either locally or joins through a provider). The *sender* is the new user object. The kwarg *from_moderation* indicates whether the user went through a moderation process (this also includes users created by an administrator manually) or was created immediately on registration; the identity associated with the registration is passed in the *identity* kwarg.
indico.core.signals.users.registration_requested
Called when a user requests to register a new indico account, i.e. if moderation is enabled. The sender is the registration request.

### 3.1.4 Adding models to your plugin

Plugins must describe its database model the in the *models* folder if needed:

```python
class Foo(db.Model):
    __tablename__ = 'foo'
    __table_args__ = {'schema': 'plugin_example'}

    id = db.Column(
        db.Integer,
        primary_key=True
    )
    bar = db.Column(
        db.String,
        nullable=False,
        default=''
    )
    location_id = db.Column(
        db.Integer,
        db.ForeignKey('roombooking.locations.id'),
        nullable=False
    )
    location = db.relationship(
        'Location',
        backref=db.backref('example_foo', cascade='all, delete-orphan', lazy='dynamic'),
    )

    @return_ascii
    def __repr__(self):
        return u'<Foo(id={}, bar={}, location={})>'.format(self.id, self.bar, self.location)
```

Thanks to Alembic, the migration needed to create the tables in the database can also be included in the plugin. The steps to do so are:

1. Create a revision for the changes your plugin will add with `indico db --plugin example migrate -m 'short description'`
2. Fine-tune the revision file generated under *migrations*.
3. Run `indico db --plugin example upgrade` to have Alembic upgrade your DB with the changes.
Indico allows you to programmatically access the content of its database by exposing various information like category contents, events, rooms and room bookings through a web service, the HTTP Export API.

4.1 Indico - HTTP API

Indico allows you to programmatically access the content of its database by exposing various information like category contents, events, rooms and room bookings through a web service, the HTTP Export API.

4.1.1 Accessing the API

URL structure

Indico allows you to programmatically access the content of its database by exposing various information like category contents, events, rooms and room bookings through a web service, the HTTP Export API.

The basic URL looks like:

http://my.indico.server/export/WHAT/{loc}/ID.TYPE?PARAMS&ak=KEY&timestamp=TS&signature=SIG

where:

- **WHAT** is the element you want to export (one of *categ, event, room, reservation*)
- **LOC** is the location of the element(s) specified by **ID** and only used for certain elements, for example, for the room booking (https://indico.server/export/room/CERN/120.json?ak=0...)
- **ID** is the ID of the element you want to export (can be a - separated list). As for example, the 120 in the above URL.
- **TYPE** is the output format (one of *json,jsonp,xml,html,ics,atom,bin*)
- **PARAMS** are various parameters affecting (filtering, sorting, ...) the result list
- **KEY, TS, SIG** are part of the *API Authentication*. 
Some examples could be:

- Export data about events in a category: https://my.indico/export/categ/2.json?from=today&to=today&pretty=yes
- Export data about an event: https://indico.server/export/event/137346.json?occ=yes&pretty=yes
- Export data about rooms: https://indico.server/export/room/CERN/120.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes
- Export your reservations: https://indico.server/export/reservation/CERN.json?ak=00000000-0000-0000-0000-000000000000&detail=reservations&from=today&to=today&bookedfor=USERNAME&pretty=yes

See more details about querying in Exporters.

**API Authentication**

**General**

The HTTP Export API uses an API key and - depending on the config - a cryptographic signature for each request.

To create an API key, go to My Profile » HTTP API and click the Create API key button. This will create an API Key and a Secret Key (if signatures are required).

It is recommended to always use the highest security level. That means if only an API key is available always include it and if a secret key is available, always sign your requests. Since you might want to retrieve only public information (instead of everything visible to your Indico user) you can add the param onlypublic=yes to the query string.

It is also possible to re-use the existing Indico session. This only makes sense if your browser accesses the API, e.g. because you are developing on Indico and want to access the API via an AJAX request. Additionally this method of authentication is restricted to GET requests. To use it, add cookieauth=yes to the query string and do not specify an API key, timestamp or signature. To prevent data leakage via CSRF the CSRF token of the current session needs to be provided as a GET argument csrftoken or a HTTP header X-CSRF-Token.

**Request Signing**

To sign a request, you need the following:

- The requested path, e.g. /export/categ/123.json
- Any additional params, e.g. limit=10
- The current UNIX timestamp
- You API key and secret key

1) Add your API key to the params (limit=10&ak=your-api-key)

2) Add the current timestamp to the params (limit=10&ak=your-api-key&timestamp=1234567890)

3) Sort the query string params (ak=your-api-key&limit=10&timestamp=1234567890)

4) Merge path and the sorted query string to a single string (/export/categ/123.json?ak=your-api-key&limit=10&timestamp=1234567890)

5) Create a HMAC-SHA1 signature of this string using your secret key as the key.

6) Append the hex-encoded signature to your query string: ?ak=your-api-key&limit=10&timestamp=1234567890&signature=your-signature
Note that a signed request might be valid only for a few seconds or minutes, so you need to sign it right before sending it and not store the generated URL as it is likely to expire soon.

You can find example code for Python and PHP in the following sections.

If persistent signatures are enabled, you can also omit the timestamp. In this case the URL is valid forever. When using this feature, please make sure to use these URLs only where necessary - use timestamped URLs whenever possible.

### Request Signing for Python

A simple example in Python:

```python
import hashlib
import hmac
import time

try:
    from urllib.parse import urlencode
except ImportError:
    from urllib import urlencode

def build_indico_request(path, params, api_key=None, secret_key=None, only_public=False, persistent=False):
    items = list(params.items())
    if hasattr(params, 'items') else list(params)
    if api_key:
        items.append(('apikey', api_key))
    if only_public:
        items.append(('onlypublic', 'yes'))
    if secret_key:
        if not persistent:
            items.append(('timestamp', str(int(time.time()))))
        items = sorted(items, key=lambda x: x[0].lower())
        url = '%s?%s' % (path, urlencode(items))
        signature = hmac.new(secret_key.encode('utf-8'), url.encode('utf-8'), hashlib.sha1).hexdigest()
        items.append(('signature', signature))
    if not items:
        return
    path
    return '%s?%s' % (path, urlencode(items))

if __name__ == '__main__':
    API_KEY = '00000000-0000-0000-0000-000000000000'
    SECRET_KEY = '00000000-0000-0000-0000-000000000000'
    PATH = '/export/categ/1337.json'
    PARAMS = {
        'limit': 123
    }
    print(build_indico_request(PATH, PARAMS, API_KEY, SECRET_KEY))
```

### Request Signing for PHP

A simple example in PHP:

```php
4.1. Indico - HTTP API
```
<?php

function build_indico_request($path, $params, $api_key = null, $secret_key = null,
    $only_public = false, $persistent = false) {
    if($api_key) {
        $params['api_key'] = $api_key;
    }

    if($only_public) {
        $params['onlypublic'] = 'yes';
    }

    if($secret_key) {
        if(!$persistent) {
            $params['timestamp'] = time();
        }
        uksort($params, 'strcasecmp');
        $url = $path . '?' . http_build_query($params);
        $params['signature'] = hash_hmac('sha1', $url, $secret_key);
    }

    if(!$params) {
        return $path;
    }

    return $path . '?' . http_build_query($params);
}

if(true) { // change to false if you want to include this file
    $API_KEY = '00000000-0000-0000-0000-000000000000';
    $SECRET_KEY = '00000000-0000-0000-0000-000000000000';
    $PATH = '/export/categ/1337.json';
    $PARAMS = array(
        'limit' => 123
    );
    echo build_indico_request($PATH, $PARAMS, $API_KEY, $SECRET_KEY) . "\n";
}

### 4.1.2 Common Parameters

The following parameters are valid for all requests no matter which element is requested. If a parameter has a shorter form, it’s given in parentheses.
<table>
<thead>
<tr>
<th>Param</th>
<th>Short</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from/to</td>
<td>f/t</td>
<td>Accepted formats:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ISO 8601 subset - YYYY-MM-DD[THH:MM]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ‘today’, ‘yesterday’, ‘tomorrow’ and ‘now’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• days in the future/past: ‘[+/-]DdHHhMMm’</td>
</tr>
<tr>
<td>pretty</td>
<td>p</td>
<td>Pretty-print the output. When exporting as JSON it will include whitespace to make the json more human-readable.</td>
</tr>
<tr>
<td>onlypublic</td>
<td>op</td>
<td>Only return results visible to unauthenticated users when set to yes.</td>
</tr>
<tr>
<td>onlyauthed</td>
<td>oa</td>
<td>Fail if the request is unauthenticated for any reason when this is set to yes.</td>
</tr>
<tr>
<td>cookieauth</td>
<td>ca</td>
<td>Use the Indico session cookie to authenticate instead of an API key.</td>
</tr>
<tr>
<td>limit</td>
<td>n</td>
<td>Return no more than the X results.</td>
</tr>
<tr>
<td>offset</td>
<td>O</td>
<td>Skip the first X results.</td>
</tr>
<tr>
<td>detail</td>
<td>d</td>
<td>Specify the detail level (values depend on the exported element)</td>
</tr>
<tr>
<td>order</td>
<td>o</td>
<td>Sort the results. Must be one of id, start, end, title.</td>
</tr>
<tr>
<td>descending</td>
<td>c</td>
<td>Sort the results in descending order when set to yes.</td>
</tr>
<tr>
<td>tz</td>
<td>-</td>
<td>Assume given timezone (default UTC) for specified dates. Example: Europe/Lisbon.</td>
</tr>
</tbody>
</table>

### 4.1.3 API Resources

#### Categories

**URL Format**

```
/export/categ/ID.TYPE
```

The ID can be either a single category ID or a - separated list. In an authenticated request the special ID favorites will be resolved to the user's list of favorites.
Parameters

<table>
<thead>
<tr>
<th>Param</th>
<th>Short</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>l</td>
<td>Only include events taking place at the specified location. The * and ? wildcards may be used.</td>
</tr>
<tr>
<td>room</td>
<td>r</td>
<td>Only include events taking place in the specified room. The * and ? wildcards may be used.</td>
</tr>
<tr>
<td>type</td>
<td>T</td>
<td>Only include events of the specified type. Must be one of: simple_event (or lecture), meeting, conference</td>
</tr>
</tbody>
</table>

Detail Levels

events

Returns basic data about the events in the category.

This is the result of the following the query https://my.indico/export/categ/2.json?from=today&to=today&pretty=yes:

```json
{
    "count": 2,
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https://my.indico/export/categ/2.json?from=today&to=today&pretty=yes",
    "ts": 1308841641,
    "results": [
        {
            "category": "TEST Category",
            "startDate": {
                "date": "2011-06-17",
                "tz": "Europe/Zurich",
                "time": "08:00:00"
            },
            "_type": "Conference",
            "endDate": {
                "date": "2011-06-30",
                "tz": "Europe/Zurich",
                "time": "18:00:00"
            },
            "description": "",
            "title": "Test EPayment",
            "url": "http://pcituds07.cern.ch/indico/conferenceDisplay.py?confId=137344",
            "location": "CERN",
            "_fossil": "conferenceMetadata",
            "timezone": "Europe/Zurich",
            "type": "conference",
            "id": "137344",
            "room": "1-1-025"
        },
        {
            "category": "TEST Category",
            "startDate": {
                "date": "2011-06-23",
                "tz": "Europe/Zurich",
                "time": "08:00:00"
            }
        }
    ]
}
```

(continues on next page)
Events

URL Format

/\export/event\ID\TYPE

The ID can be either a single event ID or a - separated list.

Parameters

<table>
<thead>
<tr>
<th>Param</th>
<th>Short</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>occurrences</td>
<td>occ</td>
<td>Include the daily event times in the exported data.</td>
</tr>
</tbody>
</table>

Detail Levels

events

Returns basic data about the event. In this example occurrences are included, too.

Result for https://indico.server/export/event/137346.json?occ=yes&pretty=yes:

```json
{
    "count": 1,
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https://indico.server/export/event/137346.json?occ=yes&pretty=yes",
    "ts": 1308899256,
    "results": [
        {
            "category": "TEST Category",
```
"startDate": {
    "date": "2011-06-23",
    "tz": "Europe/Zurich",
    "time": "08:00:00"
},
"_type": "Conference",
"endDate": {
    "date": "2011-06-24",
    "tz": "Europe/Zurich",
    "time": "18:00:00"
},
"description": "",
"title": "Export Test",
"url": "http://indico.server/conferenceDisplay.py?confId=137346",
"room": null,
"occurrences": [
    {
        "_fossil": "period",
        "endDT": {
            "date": "2011-06-23",
            "tz": "Europe/Zurich",
            "time": "08:40:00"
        },
        "startDT": {
            "date": "2011-06-23",
            "tz": "Europe/Zurich",
            "time": "08:00:00"
        },
        "_type": "Period"
    },
    {
        "_fossil": "period",
        "endDT": {
            "date": "2011-06-24",
            "tz": "Europe/Zurich",
            "time": "15:00:00"
        },
        "startDT": {
            "date": "2011-06-24",
            "tz": "Europe/Zurich",
            "time": "12:00:00"
        },
        "_type": "Period"
    }
],
"_fossil": "conferenceMetadata",
"timezone": "Europe/Zurich",
"type": "meeting",
"id": "137346",
"location": "CERN"
contributions

Includes the contributions of the event.

Output for https://indico.server/export/event/137346.json?detail=contributions&pretty=yes:

```json
{
  "count": 1,
  "_type": "HTTPAPIResult",
  "complete": true,
  "url": "https://indico.server/export/event/137346.json?detail=contributions&pretty=yes",
  "ts": 1308899252,
  "results": [
    {
      "category": "TEST Category",
      "startDate": {
        "date": "2011-06-23",
        "tz": "Europe/Zurich",
        "time": "08:00:00"
      },
      "_type": "Conference",
      "endDate": {
        "date": "2011-06-24",
        "tz": "Europe/Zurich",
        "time": "18:00:00"
      },
      "description": "",
      "title": "Export Test",
      "url": "http://indico.server/conferenceDisplay.py?confId=137346",
      "type": "meeting",
      "location": "CERN",
      "_fossil": "conferenceMetadataWithContribs",
      "timezone": "Europe/Zurich",
      "contributions": [
        {
          "startDate": {
            "date": "2011-06-23",
            "tz": "Europe/Zurich",
            "time": "08:20:00"
          },
          "_type": "Contribution",
          "endDate": {
            "date": "2011-06-23",
            "tz": "Europe/Zurich",
            "time": "08:40:00"
          },
          "description": "",
          "title": "d1c2",
          "track": null,
          "duration": 20,
          "session": null,
          "location": "CERN",
          "_fossil": "contributionMetadata",
          "type": null,
          "id": "1",
          "room": null
        }
      ]
    }
  ]
}
```

(continues on next page)
```json
{
    "startDate": {
        "date": "2011-06-23",
        "tz": "Europe/Zurich",
        "time": "08:00:00"
    },
    "_type": "Contribution",
    "endDate": {
        "date": "2011-06-23",
        "tz": "Europe/Zurich",
        "time": "08:20:00"
    },
    "description": "",
    "title": "d1c1",
    "track": null,
    "duration": 20,
    "session": null,
    "location": "CERN",
    "_fossil": "contributionMetadata",
    "type": null,
    "id": "0",
    "room": null
},
{
    "startDate": {
        "date": "2011-06-24",
        "tz": "Europe/Zurich",
        "time": "14:00:00"
    },
    "_type": "Contribution",
    "endDate": {
        "date": "2011-06-24",
        "tz": "Europe/Zurich",
        "time": "14:20:00"
    },
    "description": "",
    "title": "d2s1c1",
    "track": null,
    "duration": 20,
    "session": "d2s1",
    "location": "CERN",
    "_fossil": "contributionMetadata",
    "type": null,
    "id": "3",
    "room": null
},
{
    "startDate": {
        "date": "2011-06-24",
        "tz": "Europe/Zurich",
        "time": "12:00:00"
    },
    "_type": "Contribution",
    "endDate": {
        "date": "2011-06-24",
        "tz": "Europe/Zurich",
        "time": "14:00:00"
    },
    "description": "",
    "title": "d2s1c1",
    "track": null,
    "duration": 20,
    "session": "d2s1",
    "location": "CERN",
    "_fossil": "contributionMetadata",
    "type": null,
    "id": "3",
    "room": null
}
```
subcontributions

Like contributions, but inside the contributions the subcontributions are included in a field named subContributions.

sessions

Includes details about the different sessions and groups contributions by sessions. The top-level contributions list only contains contributions which are not assigned to any session. Subcontributions are included in this details level, too.

For example, https://indico.server/export/event/137346.json?detail=sessions&pretty=yes:

```json
{
  "count": 1,
  "_type": "HTTPAPIResult",
  "complete": true,
  "url": "https://indico.server/export/event/137346.json?detail=sessions&pretty=yes",
  "ts": 1308899771,
  "results": [
    {
      "category": "TEST Category",
      "startDate": {
        "date": "2011-06-23",
        "tz": "Europe/Zurich",
        "time": "08:00:00"
      },
      "_type": "Conference",
      "endDate": {
        "date": "2011-06-24",
        "tz": "Europe/Zurich",
        "time": "18:00:00"
      },
      "description": "",
      "title": "Export Test",
      "url": "http://indico.server/conferenceDisplay.py?confId=137346",
    }
  ]
}
```
"contributions": [
  {
    "startDate": {
      "date": "2011-06-23",
      "tz": "Europe/Zurich",
      "time": "08:20:00"
    },
    "_type": "Contribution",
    "endDate": {
      "date": "2011-06-23",
      "tz": "Europe/Zurich",
      "time": "08:40:00"
    },
    "description": "",
    "subContributions": [],
    "title": "d1c2",
    "track": null,
    "duration": 20,
    "session": null,
    "location": "CERN",
    "_fossil": "contributionMetadataWithSubContribs",
    "type": null,
    "id": "1",
    "room": null
  },
  {
    "startDate": {
      "date": "2011-06-23",
      "tz": "Europe/Zurich",
      "time": "08:00:00"
    },
    "_type": "Contribution",
    "endDate": {
      "date": "2011-06-23",
      "tz": "Europe/Zurich",
      "time": "08:20:00"
    },
    "description": "",
    "subContributions": [],
    "title": "d1c1",
    "track": null,
    "duration": 20,
    "session": null,
    "location": "CERN",
    "_fossil": "contributionMetadataWithSubContribs",
    "type": null,
    "id": "0",
    "room": null
  },
  {
    "startDate": {
      "date": "2011-06-24",
      "tz": "Europe/Zurich",
      "time": "12:00:00"
    },
    "_type": "Contribution",
    "endDate": {
      "date": "2011-06-24",
      "tz": "Europe/Zurich",
      "time": "12:40:00"
    },
    "description": "",
    "subContributions": [],
    "title": "d1c3",
    "track": null,
    "duration": 20,
    "session": null,
    "location": "CERN",
    "_fossil": "contributionMetadataWithSubContribs",
    "type": null,
    "id": "2",
    "room": null
  }
]
"date": "2011-06-24",
"tz": "Europe/Zurich",
"time": "14:00:00"
},
"description": "",
"subContributions": [
],
"title": "d2c1",
"track": null,
"duration": 120,
"session": null,
"location": "CERN",
"_fossil": "contributionMetadataWithSubContribs",
"type": null,
"id": "2",
"room": null
},
"sessions": [
{
"startDate": {
"date": "2011-06-24",
"tz": "Europe/Zurich",
"time": "14:00:00"
},
"_type": "Session",
"room": "",
"numSlots": 1,
"color": "#EEE0EF",
"material": [],
"isPoster": false,
"sessionConveners": [],
"location": "CERN",
"address": "",
"_fossil": "sessionMetadata",
"title": "d2s1",
"textColor": "#1D041F",
"contributions": [
{
"startDate": {
"date": "2011-06-24",
"tz": "Europe/Zurich",
"time": "14:00:00"
},
"_type": "Contribution",
"endDate": {
"date": "2011-06-24",
"tz": "Europe/Zurich",
"time": "14:20:00"
},
"description": "",
"subContributions": [],
"title": "d2s1c1",
"track": null,
"duration": 20,
"session": "d2s1",
"location": "CERN",
"_fossil": "contributionMetadataWithSubContribs",
"type": null,
"id": "1",
"room": null
}]},
"type": null,
"id": "3",
"room": null

]},
"id": "0"
],
"location": "CERN",
"_fossil": "conferenceMetadataWithSessions",
"timezone": "Europe/Zurich",
"type": "meeting",
"id": "137346",
"room": null
}
]

Timetable

URL Format

/export/timetable/ID.TYPE

The ID should be the event ID, e.g. 123.

Results

Returns the timetable of the event.

Result for https://indico.server/export/timetable/137346.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes:

```json
{
   "count": 1,
   "additionalInfo": {},
   "_type": "HTTPAPIResult",
   "complete": true,
   "url": "https:\/\/indico.server\/export\/timetable\/137346.json?ak=00000000-0000-0000-000000000000&pretty=yes",
   "ts": 1367242732,
   "results": {
      "137346": {
         "20130429": {
            "c0": {
               "startDate": {
                  "date": "2013-04-29",
                  "tz": "Europe/Zurich",
                  "time": "16:00:00"
               },
               "_type": "ContribSchEntry",
               "material": [],
               "endDate": {
                  "date": "2013-04-29",
               }
            }
         }
      }
   }
}
```
Event Search

URL Format

/export/event/search/TERM.TYPE

The TERM should be a string, e.g. “ichep”

Results

Returns the events found.

Result for https://indico.server/export/event/search/ichep.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes:

```json
{
  "count": 5,
  "additionalInfo": {},
  "_type": "HTTPAPIResult",
  "complete": true,
  "url": "https:\/\slash\/indico.server\slash\/export\slash\/event\slash\slash\/search\slash\/ichep.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes",
  "ts": 1367245058,
  "results": [
    {
      "startDate": {
        "date": "2010-07-16",
        "tz": "UTC",
        "time": "11:00:00"
      },
      "hasAnyProtection": false,
      "id": "101465",
      "title": "Rehearsals for ICHEP Friday 16th July Afternoon Session"
    }
  ]
}
```
Files

General Information

The file export is only available for authenticated users, i.e. when using an API key and a signature (if enabled).
**URL Format**

/export/event/EVENT_ID/session/SESSION_ID/contrib/CONTRIBUTION_ID/subcontrib/SUBCONTRIBUTION_ID/material/MATERIAL_ID/RESOURCE_ID.TYPE

All ID’s should be single ID, not separated list.

The EVENT_ID should be the event ID, e.g. 123.
The SESSION_ID (optional) should be the session ID, e.g. 4.
The CONTRIBUTION_ID (optional) should be the contribution ID, e.g. 3.
The SUBCONTRIBUTION_ID (optional) should be the sub-contribution ID, e.g. 1.
The MATERIAL_ID should by the material name if it came default group e.g. Slides or material ID if not, e.g. 2.
The RESOURCE_ID should by the resource ID.
Only supported TYPE for files is bin (binary data).

**Parameters**

None

**Detail Levels**

**file**

Returns file (or an error in JSON format).

For example:  
https://indico.server/export/event/23/session/0/contrib/3/material/slides/3.bin?ak=00000000-0000-0000-0000-000000000000

**User**

**General Information**

The user export is only available for authenticated users, i.e. when using an API key and a signature (if enabled).

**URL Format**

/export/user/USER_ID.TYPE

The USER_ID should be the user ID, e.g. 44.

**Parameters**

None
Results

Returns the user information (or an error in JSON format).

Result for https://indico.server/export/user/36024.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes:

```json
{
    "count": 1,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https:\/\/indico.server\/export\/user\/36024.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes",
    "ts": 1367243741,
    "results": [
        {
            "_type": "Avatar",
            "name": "Alberto RESCO PEREZ",
            "firstName": "Alberto",
            "affiliation": "CERN",
            "familyName": "Resco Perez",
            "email": "test@cern.ch",
            "phone": "+41XXXXXXXX",
            "_fossil": "avatar",
            "title": ",",
            "id": "36024"
        }
    ]
}
```

Room Booking

Bookings

Creating bookings

General Information

The Room Booking API is only available for authenticated users, i.e. when using an API key and a signature (if enabled). If the room booking system is restricted to certain users/groups this restriction applies for this API, too. The request will fail if there is a collision with another booking, blocking or unavailable period.

Note that it is not possible to pre-book a room through this api.

URL Format

/api/roomBooking/bookRoom.TYPE

TYPE should be json or xml.

Parameters

The following parameters are required:
<table>
<thead>
<tr>
<th>Param</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>text</td>
<td>Room location, e.g. CERN</td>
</tr>
<tr>
<td>roomid</td>
<td>text</td>
<td>Room id</td>
</tr>
<tr>
<td>from/to</td>
<td>f/t</td>
<td>Start/End time for a booking. Accepted formats:</td>
</tr>
</tbody>
</table>
|         |        |   - ISO 8601 subset - YYYY-MM-DD[THH:MM]  
|         |        |   - ‘today’, ‘yesterday’, ‘tomorrow’ and ‘now’  
|         |        |   - days in the future/past: ‘[+/-]DdHHhMMm’  |
| reason  | text   | Reason for booking a room           |
| username| text   | User login name for whom the booking will be created |

### Booking a room

**POST request**

Returns **reservation id** if the booking was successful or error information if there were any problems.

For example:

```
curl --data "username=jdoe&from=2012-12-30T21:30&to=2012-12-30T22:15&reason=meeting&location=CERN&roomid=189" 'http://indico.server/indico/api/roomBooking/bookRoom.json'
```

Result:

```
{
  "url": "\api\roomBooking\bookRoom.json",
  "_type": "HTTPAPIResult",
  "results": {
    "reservationID": 45937
  },
  "ts": 1354695663
}
```

### Retrieving bookings

#### General Information

The reservation export is only available for authenticated users, i.e. when using an API key and a signature (if enabled). If the room booking system is restricted to certain users/groups this restriction applies for the reservation export API, too.

Please note that the room export with the **reservations** detail level is much more appropriate if you need reservations for specific rooms.

---

4.1. Indico - HTTP API
URL Format

/export/reservation/LOCATION.TYPE

The LOCATION should be the room location, e.g. CERN. A - separated list of multiple locations is allowed, too.

Parameters

<table>
<thead>
<tr>
<th>Param</th>
<th>Short</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>occurrences</td>
<td>occ</td>
<td>yes, no</td>
<td>Include all occurrences of room reservations.</td>
</tr>
<tr>
<td>cancelled</td>
<td>cxl</td>
<td>yes, no</td>
<td>If specified only include cancelled (yes) or non-cancelled (no) reservations.</td>
</tr>
<tr>
<td>rejected</td>
<td>rej</td>
<td>yes, no</td>
<td>If specified only include rejected/non-rejected resvs.</td>
</tr>
<tr>
<td>confirmed</td>
<td>-</td>
<td>yes, no pending</td>
<td>If specified only include bookings/pre-bookings with the given state.</td>
</tr>
<tr>
<td>archival</td>
<td>arch</td>
<td>yes, no</td>
<td>If specified only include bookings (not) from the past.</td>
</tr>
<tr>
<td>recurring</td>
<td>rec</td>
<td>yes, no</td>
<td>If specified only include bookings which are (not) recurring.</td>
</tr>
<tr>
<td>repeating</td>
<td>rep</td>
<td>yes, no</td>
<td>Alias for recurring</td>
</tr>
<tr>
<td>booked-for</td>
<td>bf</td>
<td>text (wildcards)</td>
<td>Only include bookings where the booked for field matches the given wildcard string.</td>
</tr>
<tr>
<td>occurs</td>
<td>-</td>
<td>yyyy-mm-dd</td>
<td>Only include bookings which have a valid occurrence on the given date. Multiple dates can be separated by commas.</td>
</tr>
</tbody>
</table>

Detail Levels

reservations

Returns detailed data about the reservations and the most important information about the booked room.

For example, https://indico.server/export/reservation/CERN.json?ak=00000000-0000-0000-0000-000000000000&detail=reservation&from=today&to=today&pretty=yes:

```json
{
    "count": 1,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "url": "/export/reservation/CERN.json?ak=00000000-0000-0000-0000-000000000000&detail=reservation&from=today&to=today&pretty=yes",
    "results": [
        {
            "_type": "Reservation",
            "repeat_unit": 1,
            "endDT": {
                "date": "2014-08-14",
                "tz": "Europe/Zurich",
                "time": "12:30:00"
            },
            "room": {
                "_type": "Room",
                "..."  
            }
        }
    ]
}
```

(continues on next page)
Rooms

General Information

The room export is only available for authenticated users, i.e. when using an API key and a signature (if enabled). If the room booking system is restricted to certain users/groups this restriction applies for the room export API, too.

URL Format

/export/room/LOCATION/ID.TYPE

The LOCATION should be the room location, e.g. CERN. The ID can be either a single room ID or a separated list.
### Parameters

<table>
<thead>
<tr>
<th>Param</th>
<th>Short</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>occurrences</td>
<td>occ</td>
<td>yes, no</td>
<td>Include all occurrences of room reservations.</td>
</tr>
<tr>
<td>cancelled</td>
<td>cxl</td>
<td>yes, no</td>
<td>If specified only include cancelled (yes) or non-cancelled (no) reservations.</td>
</tr>
<tr>
<td>rejected</td>
<td>rej</td>
<td>yes, no</td>
<td>If specified only include rejected/non-rejected resvs.</td>
</tr>
<tr>
<td>confirmed</td>
<td>rec</td>
<td>yes, no, pending</td>
<td>If specified only include bookings/pre-bookings with the given state.</td>
</tr>
<tr>
<td>archival</td>
<td>arch</td>
<td>yes, no</td>
<td>If specified only include bookings (not) from the past.</td>
</tr>
<tr>
<td>recurring</td>
<td>rep</td>
<td>yes, no</td>
<td>If specified only include bookings which are (not) recurring.</td>
</tr>
<tr>
<td>booked-for</td>
<td>bf</td>
<td>text (wildcards)</td>
<td>Only include bookings where the booked for field matches the given wildcard string.</td>
</tr>
<tr>
<td>occurs</td>
<td>-</td>
<td>yyyy-mm-dd</td>
<td>Only include bookings which have a valid occurrence on the given date. Multiple dates can be separated by commas.</td>
</tr>
</tbody>
</table>

### Detail Levels

**rooms**

Returns basic data about the rooms.

For example, `https://indico.server/export/room/CERN/57.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes`:

```json
{
    "count": 1,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "url": "/export/room/CERN/57.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes",
    "results": [
        {
            "building": "500",
            "_type": "Room",
            "name": "Main Auditorium",
            "floor": "1",
            "longitude": "6.0542704900999995",
            "vcList": [
                "Audio Conference",
                "Built-in (MCU) Bridge",
                "CERN MCU",
                "ESnet MCU",
                "EVO",
                "H323 point2point",
                "Vidyo"
            ],
            "equipment": [
                "Blackboard",
            ]
        }
    ]
}
```

(continues on next page)
reservations

Returns basic data about the rooms and their reservations in the given timeframe.

Output for https://indico.server/export/room/CERN/57.json?ak=00000000-0000-0000-0000-000000000000&detail=reservations&from=today&to=today&pretty=yes:

```json
{
  "count": 1,
  "additionalInfo": {},
  "_type": "HTTPAPIResult",
  "url": "/export/room/CERN/57.json?ak=00000000-0000-0000-0000-000000000000&detail=reservations&from=today&to=today&pretty=yes",
  "results": [
  {
    "building": "500",
    "_type": "Room",
    "name": "Main Auditorium",
    "floor": "1",
    "reservations": [
      {
        "_type": "Reservation",
        "repeat_unit": 1,
        "endDT": {
          "date": "2014-08-14",
          "tz": "Europe/Zurich",
          "time": "12:30:00"
        },
        "isConfirmed": true,
        "isValid": true,
        "repeatability": "daily",
        "repeat_step": 1,
        "vList": [],
        "reason": "Summer Student Lecture programme",
        "bookedForName": "DOE, John",
      }
    ]
  }
}
```

(continues on next page)
"is_rejected": false,
"is_cancelled": false,
"startDT": {
  "date": "2014-07-02",
  "tz": "Europe/Zurich",
  "time": "08:30:00"
},
"id": 63779,
"bookingUrl": "http://pcavc005.cern.ch:8000/indico/rooms/booking/CERN/63779/",
"location": "CERN"
"longitude": "6.0542704900999995",
"vcList": [
  "Audio Conference",
  "Built-in (MCU) Bridge",
  "CERN MCU",
  "ESnet MCU",
  "EVO",
  "H323 point2point",
  "Vidyo"
],
"equipment": [
  "Blackboard",
  "Computer Projector",
  "Ethernet",
  "Microphone",
  "PC",
  "Telephone conference",
  "Video conference",
  "Webcast/Recording",
  "Wireless"
],
"roomNr": "001",
"location": "CERN",
"latitude": "46.23141394580001",
"fullName": "500-1-001 - Main Auditorium",
"id": 57,
"bookingUrl": "/indico/rooms/room/CERN/57/book"
],
"ts": 1406731966
}
The *LOCATION* should be the room location, e.g. *CERN*. The *ROOMNAME* is a single ROOMNAME.

**Parameters**

No parameters needed.

**Results**

Returns basic data about the rooms.

For example, https://indico.server/export/roomName/CERN/Main Auditorium.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes:

```json
{
   "count": 1,
   "additionalInfo": {},
   "_type": "HTTPAPIResult",
   "url": "/export/roomName/CERN/Main Auditorium.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes",
   "results": [
      {
         "building": "500",
         "_type": "Room",
         "name": "Main Auditorium",
         "floor": "1",
         "longitude": "6.0542704900999995",
         "vcList": [
            "Audio Conference",
            "Built-in (MCU) Bridge",
            "CERN MCU",
            "ESnet MCU",
            "EVO",
            "H323 point2point",
            "Vidyo"
         ],
         "equipment": [
            "Blackboard",
            "Computer Projector",
            "Ethernet",
            "Microphone",
            "PC",
            "Telephone conference",
            "Video conference",
            "Webcast/Recording",
            "Wireless"
         ],
         "roomNr": "001",
         "location": "CERN",
         "latitude": "46.23141394580001",
         "fullName": "500-1-001 - Main Auditorium",
         "id": 57,
         "bookingUrl": "/indico/rooms/room/CERN/57/book"
      }
   ],
   "ts": 1406732578
}
```
4.1.4 HTTP API Tools
This part of the documentation focuses on the core modules of Indico and includes information about the models and utility functions and classes that are useful for understanding the internals of the application.

5.1 API reference

This part of the documentation focuses on the core modules of Indico and includes information about the models and utility functions and classes that are useful for understanding the internals of the application.

5.1.1 Event

Todo: Docstrings (module, models, operations, utilities, settings)

Models

class indico.modules.events.models.events.Event(**kwargs)
    Bases: indico.core.db.sqlalchemy.searchable_titles.SearchableTitleMixin, indico.core.db.sqlalchemy.descriptions.DescriptionMixin, indico.core.db.sqlalchemy.locations.LocationMixin, indico.core.db.sqlalchemy.protection.ProtectionManagersMixin, indico.core.db.sqlalchemy.attachments.AttachedItemsMixin, indico.core.db.sqlalchemy.notes.AttachedNotesMixin, indico.modules.events.models.persons.PersonLinkDataMixin, sqlalchemy.ext.declarative.api.Model

An Indico event

This model contains the most basic information related to an event.

Note that the ACL is currently only used for managers but not for view access!
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

\[
\text{ATTACHMENT_FOLDER_ID_COLUMN} = \text{u'event_id'}
\]

\text{access_key}

\text{acl_entries}

  The ACL entries for the event

\text{additional_info}

\text{allow_access_key} = \text{True}

\text{allow_location_inheritance} = \text{False}

\text{allow_no_access_contact} = \text{True}

\text{can_lock} (\text{user})

  Check whether the user can lock/unlock the event

\text{category}

  The category containing the event

\text{classmethod category_chain_overlaps} (\text{category_ids})

  Create a filter that checks whether the event has any of the provided category ids in its parent chain.

\text{Parameters category_ids} – A list of category ids or a single category id

\text{category_id}

  The ID of immediate parent category of the event

\text{cfa}

\text{cfp}

\text{cloned_from}

  The event this one was cloned from

\text{cloned_from_id}

  If this event was cloned, the id of the parent event

\text{contact_emails}

\text{contact_phones}

\text{contact_title}

\text{contribution_types}

\text{created_dt}

  The creation date of the event

\text{creator}

  The user who created the event

\text{creator_id}

  The ID of the user who created the event

\text{default_page}

  The event’s default page (conferences only)
**default_page_id**
The ID of the event’s default page (conferences only)

**default_render_mode = 1**

**delete (reason, user=None)**

**disallowed_protection_modes = frozenset([])**

**display_tzinfo**
The tzinfo of the event as preferred by the current user

**duration**

**end_dt**
The end date of the event

**end_dt_display**
The ‘displayed end dt’, which is usually the actual end dt, but may be overridden for a conference.

**end_dt_local**

**end_dt_override**

**ends_after (dt)**
Check whether the event ends on/after the specified date

**event**
Convenience property so all event entities have it

**external_logo_url**

**external_url**

**get_allowed_sender_emails (include_current_user=True, include_creator=True, include_managers=True, include_contact=True, include_chairs=True, extra=None)**
Return the emails of people who can be used as senders (or rather Reply-to contacts) in emails sent from within an event.

**Parameters**

- **include_current_user** – Whether to include the email of the currently logged-in user
- **include_creator** – Whether to include the email of the event creator
- **include_managers** – Whether to include the email of all event managers
- **include_contact** – Whether to include the “event contact” emails
- **include_chairs** – Whether to include the emails of event chairpersons (or lecture speakers)
- **extra** – An email address that is always included, even if it is not in any of the included lists.

**Returns** An OrderedDict mapping emails to pretty names

**get_contribution (id_)**
Get a contribution of the event

**get_contribution_field (field_id)**

**get_non_inheriting_objects ()**
Get a set of child objects that do not inherit protection
get_relative_event_ids()
    Get the first, last, previous and next event IDs.
    Any of those values may be None if there is no matching event or if it would be the current event.
    
    Returns: A dict containing first, last, prev and next.

get_session(id_=None, friendly_id=None)
    Get a session of the event

get_session_block(id_, scheduled_only=False)
    Get a session block of the event

get_sorted_tracks()
    Return tracks and track groups in the correct order

get_verbose_title(show_speakers=False, show_series_pos=False)
    Get the event title with some additional information

    Parameters
    - show_speakers – Whether to prefix the title with the speakers of the event.
    - show_series_pos – Whether to suffix the title with the position and total count in the event’s series.

happens_between(from_dt=None, to_dt=None)
    Check whether the event takes place within two dates

has Ended

has_feature(**kwargs)
    Checks if a feature is enabled for the event

has_logo

has_stylesheet

id
    The ID of the event

inherit_location = False

inheriting_have_acl = True

is_deleted
    If the event has been deleted

is_locked
    If the event is locked (read-only mode)

classmethod is_visible_in(category_id)
    Create a filter that checks whether the event is visible in the specified category.

iter_days(tzinfo=None)

keywords
    A list of tags/keywords for the event

layout_images

location_backref_name = u'events'

locator

log(realm, kind, module, summary, user=None, type_=u'simple', data=None)
    Creates a new log entry for the event
Parameters

- **realm** – A value from `EventLogRealm` indicating the realm of the action.
- **kind** – A value from `EventLogKind` indicating the kind of the action that was performed.
- **module** – A human-friendly string describing the module related to the action.
- **summary** – A one-line summary describing the logged action.
- **user** – The user who performed the action.
- **type** – The type of the log entry. This is used for custom rendering of the log message/data
- **data** – JSON-serializable data specific to the log type.

**Returns**  The newly created `EventLogEntry`

In most cases the simple log type is fine. For this type, any items from data will be shown in the detailed view of the log entry. You may either use a dict (which will be sorted) alphabetically or a list of `key`, `value` pairs which will be displayed in the given order.

**log_entries**

**logging_disabled**

Temporarily disables event logging

This is useful when performing actions e.g. during event creation or at other times where adding entries to the event log doesn’t make sense.

**logo**

The logo’s raw image data

**logo_metadata**

The metadata of the logo (hash, size, filename, content_type)

**logo_url**

**move**(category)

**move_start_dt**(start_dt)

Set event start_dt and adjust its timetable entries

**organizer_info**

**own_address**

**own_no_access_contact**

**own_room**

**own_room_id**

**own_room_name**

**own_venue**

**own_venue_id**

**own_venue_name**

**participation_regform**

**person_links**

Persons associated with this event
possible_render_modes = set([<RenderMode.html: 1>])
preloaded_all_acl_entries()
protection_mode
protection_parent
publishedRegistrations
references
    External references associated with this event
render_mode = 1
reservations
scheduled_notes
series
    The series this event is part of
series_id
    The ID of the series this events belongs to
short_external_url
short_url
start_dt
    The start date of the event
start_dt_display
    The ‘displayed start dt’, which is usually the actual start dt, but may be overridden for a conference.
start_dt_local
start_dt_override
starts_between(from_dt=None, to_dt=None)
    Check whether the event starts within two dates
static_list_links
stylesheet
    The stylesheet’s raw image data
stylesheet_metadata
    The metadata of the stylesheet (hash, size, filename)
theme
timetable_entries
timezone
    The timezone of the event
title
type
type_
tzinfo
url
url_shortcut
   The URL shortcut for the event

visibility
   The visibility depth in category overviews

class indico.modules.events.models.events.EventType
   Bases: indico.util.struct.enum.RichIntEnum
   conference = 3
   lecture = 1
   legacy_name
   meeting = 2

class indico.modules.events.models.persons.AuthorsSpeakersMixin
   Bases: object
   AUTHORS_SPEAKERS_DISPLAY_ORDER_ATTR = u'display_order_key'
   primary_authors
   secondary_authors
   speakers

class indico.modules.events.models.persons.EventPerson(**kwargs)
   Bases: indico.modules.users.models.users.PersonMixin, sqlalchemy.ext.declarative.api.Model
   A person inside an event, e.g. a speaker/author etc.
   A simple constructor that allows initialization from kwargs.
   Sets attributes on the constructed instance using the names and values in kwargs.
   Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
   address
   affiliation
   classmethod create_from_user(user, event=None, is_untrusted=False)
   email
   event
   event_id
   first_name
   classmethod for_user(user, event=None, is_untrusted=False)
      Return EventPerson for a matching User in Event creating if needed
   has_role(role, obj)
      Whether the person has a role in the ACL list of a given object
   id
   invited_dt
   is_untrusted
   last_name
classmethod link_user_by_email(user)
    Links all email-based persons matching the user’s email addresses with the user.

    Parameters
    user – A User object.

locator
    Defines a smart locator property.

    This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

    This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

    If you need more than one locator, you can define it like this:

    ```python
    @locator_property
    def locator(self):
        return {...}

    @locator.other
    def locator(self):
        return {...}
    ```

    The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

merge_person_info(**kwargs)

classmethod merge_users(target, source)
    Merge the EventPersons of two users.

    Parameters
    • target – The target user of the merge
    • source – The user that is being merged into target

phone
principal
subcontribution_links
user
user_id

class indico.modules.events.models.persons.EventPersonLink(*args, **kwargs)
    Bases: indico.modules.events.models.persons.PersonLinkBase

    Association between EventPerson and Event.

    Chairperson or speaker (lecture)

display_order
event_id
id
is_submitter
object_relationship_name = u'event'
person
person_id

person_link_backref_name = u'event_links'

person_link_unique_columns = (u'event_id',)

class indico.modules.events.models.persons.PersonLinkBase(*args, **kwargs)
    Bases: indico.modules.users.models.users.PersonMixin, sqlalchemy.ext.declarative.api.Model

    Base class for EventPerson associations.

    address

    affiliation

    display_order = Column(None, Integer(), table=None, nullable=False, default=ColumnDefault(0))

    display_order_key

    display_order_key_lastname

    email

    first_name

    id = Column(None, Integer(), table=None, primary_key=True, nullable=False)

    last_name

    object

    object_relationship_name = None

    The name of the relationship pointing to the object the person is linked to

    person = <RelationshipProperty at 0x7fcf70e1cf30; no key>

    person_id = Column(None, Integer(), ForeignKey(u'events.persons.id'), table=None, nullable=False)

    person_link_backref_name = None

    The name of the backref on the EventPerson

    person_link_unique_columns = None

    The columns which should be included in the unique constraint.

    phone

    title

class indico.modules.events.models.persons.PersonLinkDataMixin
    Bases: object

    person_link_data

class indico.modules.events.models.principals.EventPrincipal(**kwargs)
    Bases: indico.core.db.sqlalchemy.principals.PrincipalPermissionsMixin, sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in kwargs.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.

    allow_category_roles = True

    allow_emails = True

5.1. API reference
allow_event_roles = True
allow_networks = True
category_role
category_role_id
event
event_id
   The ID of the associated event
event_role
event_role_id
full_access
id
   The ID of the acl entry
ip_network_group
ip_network_group_id
local_group
local_group_id
multipass_group_name
multipass_group_provider
permissions
principal_backref_name = u'in_event_acls'
principal_for = u'Event'
read_access
type
unique_columns = (u'event_id',)
user
user_id
class indico.modules.events.models.references.EventReference(**kwargs)
   Bases: indico.modules.events.models.references.ReferenceModelBase
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.
event_id
id
reference_backref_name = u'event_references'
reference_type
reference_type_id
value

class indico.modules.events.models.references.ReferenceModelBase(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    id = Column(None, Integer(), table=None, primary_key=True, nullable=False)
    reference_backref_name = None
        The name of the backref on the ReferenceType
    reference_type = <RelationshipProperty at 0x7fcf70cb9ea8; no key>
    reference_type_id = Column(None, Integer(), ForeignKey(u'indico.reference_types.id'), table=None, nullable=False)
    url
        The URL of the referenced entity.
        None if no URL template is defined.
    urn
        The URN of the referenced entity.
        None if no scheme is defined.
    value = Column(None, String(), table=None, nullable=False)

class indico.modules.events.models.references.ReferenceType(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    event_references

    id
        The unique ID of the reference type

    locator
        Defines a smart locator property.
        This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
        This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.
        If you need more than one locator, you can define it like this:

        @locator_property
        def locator(self):
            return {...}

        @locator.other
        def locator(self):
            return {...}
The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

**name**
The name of the referenced system

**scheme**
The scheme used to build an URN for the reference

**url_template**
A URL template to build a link to a referenced entity

```python
class indico.modules.events.models.reviews.ProposalCommentMixin
    Bases: object
    can_edit (user)

timeline_item_type = u'comment'
```

```python
class indico.modules.events.models.reviews.ProposalGroupProxy (group)
    Bases: object
```

Represents the object that the proposals can be grouped by.

It provides all necessary methods for building the URLs, displaying the grouping information, etc.

**full_title**

```
full_title_attr = u'full_title'
```

**locator**
Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

**title**

```
title_attr = u'title'
```

```python
class indico.modules.events.models.reviews.ProposalMixin
    Bases: object
```

Classes that represent a proposal object should extend this class (ex: Abstract, Paper).

**call_for_proposals_attr = None**
Attribute to retrieve the object with access to the reviewing settings

```python
can_comment (user)
```
can_review (user, check_state=False)
cfp
create_comment_endpoint = None
create_judgment_endpoint = None
create_review_endpoint = None
delete_comment_endpoint = None
edit_comment_endpoint = None
edit_review_endpoint = None
get_delete_comment_url (comment)
get_last_revision ()
get_revisions ()
get_save_comment_url (comment=None)
get_save_judgment_url ()
get_save_review_url (group=None, review=None)
is_in_final_state
proposal_type = None
    A unique identifier to handle rendering differences between proposal types
revisions_enabled = True
    Whether there is support for multiple revisions per proposal or just one
class indico.modules.events.models.reviews.ProposalReviewMixin
    Bases: object
    Mixin for proposal reviews
    Classes that represent a review of a proposal should extend this class (ex: AbstractReview, PaperReview).
can_edit (user)
group
group_attr = None
    Object used to group reviews together
group_proxy_cls
    Proxy class to provide the necessary properties and methods to the review grouping object
    alias of ProposalGroupProxy
revision
revision_attr = None
    The revision object that the review refers to
score
timeline_item_type = u'review'
    A unique identifier to handle rendering differences between timeline items
class indico.modules.events.models.reviews.ProposalRevisionMixin
    Bases: object
    Properties and methods of a proposal revision.
Indico Documentation, Release 2.3-dev

get_reviewed_for_groups(user, include_reviewed=False)

get_reviewer_render_data(**kwargs)

get_reviews(group=None, user=None)

get_timeline(user=None)

proposal

proposal_attr = None

The attribute of the revision used to fetch the proposal object.

revisions_enabled = True

Whether the reviewing process supports multiple revisions per proposal. If set to false it is assumed that
the reviewing process supports only one revision per proposal.

class indico.modules.events.models.series.EventSeries(**kwargs)

Bases: sqlalchemy.ext.declarative.api.Model

A series of events.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.

id

The ID of the series

show_links

Whether to show links to the other events in the same series on the main event page.

show_sequence_in_title

Whether to show the sequence number of an event in its title on category display pages and on the main
event page.

class indico.modules.events.models.settings.EventSetting(**kwargs)

Bases: indico.core.settings.models.base.JSONSettingsBase, indico.modules.
events.models.settings.EventSettingsMixin, sqlalchemy.ext.declarative.api.
Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.

event

event_id

id

module

name

settings_backref_name = u'settings'

value
class indico.modules.events.models.settings.EventSettingPrincipal(**kwargs)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

allow_category_roles = True
allow_event_roles = True
category_role
category_role_id
e-mail = None
event
event_id
event_role
event_role_id
extra_key_cols = (u'event_id',)
id
ip_network_group = None
ip_network_group_id = None
local_group
local_group_id
module
multipass_group_name
multipass_group_provider
name
principal_backref_name = u'in_event_settings_acls'
settings_backref_name = u'settings_principals'
type
user
user_id

class indico.modules.events.models.settings.EventSettingsMixin
Bases: object

event = <RelationshipProperty at 0x7fcf70a8f9e0; no key>
event_id = Column(None, Integer(), ForeignKey(u'events.events.id'), table=None, nullable=False)
settings_backref_name = None
class indico.modules.events.models.static_list_links.StaticListLink(**kwargs)

Bases: sqlalchemy.ext.declarative.api.Model

Display configuration data used in static links to listing pages. This allows users to share links to listing pages in events while preserving e.g. column/filter configurations. A simple constructor that allows initialization from kwargs. Sets attributes on the constructed instance using the names and values in kwargs. Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

classmethod create(event, type_, data)
    Create a new static list link.
    If one exists with the same data, that link is used instead of creating a new one.

    Parameters
    • event – the Event for which to create the link
    • type – the type of the link
    • data – the data to associate with the link

    Returns the newly created StaticListLink

created_dt
data
event
event_id
id
last_used_dt

classmethod load(event, type_, uuid)
    Load the data associated with a link

    Parameters
    • event – the Event the link belongs to
    • type – the type of the link
    • uuid – the UUID of the link

    Returns the link data or None if the link does not exist

type
uuid

Operations

indico.modules.events.operations.clone_event(event, start_dt, cloners, category=None)
    Clone an event on a given date/time.
    Runs all required cloners.

    Parameters
    • start_dt – The start datetime of the new event;
• **cloners** – A set containing the names of all enabled cloners;

• **category** – The Category the new event will be created in.

```python
indico.modules.events.operations.create_event(*args, **kwargs)
indico.modules.events.operations.create_event_references(event, data)
indico.modules.events.operations.create_reference_type(data)
indico.modules.events.operations.create_reviewing_question(event, question_model, wtf_field_cls, form, data=None)
indico.modules.events.operations.delete_reference_type(reference_type)
indico.modules.events.operations.delete_reviewing_question(question)
indico.modules.events.operations.lock_event(event)
indico.modules.events.operations.sort_reviewing_questions(questions, new_positions)
indico.modules.events.operations.unlock_event(event)
indico.modules.events.operations.update_event(event, update_timetable=False, **data)
indico.modules.events.operations.update_event_protection(event, data)
indico.modules.events.operations.update_event_type(event, type_)
indico.modules.events.operations.update_reference_type(reference_type, data)
indico.modules.events.operations.update_reviewing_question(question, form)
```

**Utilities**

```python
class indico.modules.events.util.ListGeneratorBase(event, entry_parent=None)
    Bases: object
    Base class for classes performing actions on Indico object lists.

    Parameters
    ----------
    • **event** – The associated Event
    • **entry_parent** – The parent of the entries of the list. If it’s None, the parent is assumed to be the event itself.

default_list_config = None
    The default list configuration dictionary
endpoint = None
    The endpoint of the list management page
entry_parent = None
    The parent object of the list items
event = None
    The event the list is associated with
flash_info_message(obj)
generate_static_url()
    Return a URL with a uuid referring to the list’s configuration.
```
get_list_url (uuid=None, external=False)

Return the URL of the list management page.

list_link_type = None
Unique list identifier

static_items = None
Columns that originate from the list item’s properties, relationships etc, but not from user defined fields (e.g. registration/contribution fields)

store_configuration()
Load the filters from the request and store them in the session.

class indico.modules.events.util.ZipGeneratorMixin
Mixin for RHs that generate zip with files

indico.modules.events.util.check_event_locked (rh, event, force=False)

indico.modules.events.util.check_permissions (event, field, allow_networks=False)

indico.modules.events.util.create_event_logo_tmp_file (event, tmpdir=None)

If tmpdir is specified, the logo file is created in there and a path relative to that directory is returned.

indico.modules.events.util.get_base_ical_parameters (user, detail, path, params=None)

Returns a dict of all parameters expected by iCal template

indico.modules.events.util.get_events_created_by (user, dt=None)

Gets the IDs of events created by the user

Parameters

• user – A User
• dt – Only include events taking place on/after that date

Returns A set of event ids

indico.modules.events.util.get_events_managed_by (user, dt=None)

Gets the IDs of events where the user has management privs.

Parameters

• user – A User
• dt – Only include events taking place on/after that date

Returns A set of event ids

indico.modules.events.util.get_events_with_linked_event_persons (user, dt=None)

Returns a dict containing the event ids and role for all events where the user is a chairperson or (in case of a lecture) speaker.

Parameters

• user – A User
• dt – Only include events taking place on/after that date

indico.modules.events.util.get_field_values (form_data)

Split the form fields between custom and static
indico.modules.events.util.get_object_from_args(args=None)

Retrieves an event object from request arguments.

This utility is meant to be used in cases where the same controller can deal with objects attached to various parts of an event which use different URLs to indicate which object to use.

Parameters args – The request arguments. If unspecified, request.view_args is used.

Returns An (object_type, event, object) tuple. The event is always the Event associated with the object. The object may be an Event, Session, Contribution or SubContribution.

If the object does not exist, (object_type, None, None) is returned.

indico.modules.events.util.get_random_color(event)

indico.modules.events.util.get_theme(event, override_theme_id=None)

Get the theme ID and whether it’s an override.

This is useful for places where a user may specify a different timetable theme. If the override theme is not valid for the event, a message is flashed and an exception redirecting the user to the main event page is raised.

Raises BadRequest – if the override theme id is not valid

Returns a (theme_id, is_override) tuple

indico.modules.events.util.register_event_time_change(event)

Register a time-related change for an event

This is an internal helper function used in the model to record changes of the start time or end time. The changes are exposed through the track_time_changes contextmanager function.

indico.modules.events.util.register_time_change(entry)

Register a time-related change for a timetable entry

This is an internal helper function used in the models to record changes of the start time or duration. The changes are exposed through the track_time_changes contextmanager function.

indico.modules.events.util.serialize_event_for_ical(event, detail_level)

indico.modules.events.util.serialize_event_for_json_ld(event, full=False)

indico.modules.events.util.serialize_event_person(person)

Serialize EventPerson to JSON-like object

indico.modules.events.util.serialize_person_for_json_ld(person)

Serialize PersonLink to JSON-like object

indico.modules.events.util.set_custom_fields(obj, custom_fields_data)

indico.modules.events.util.track_time_changes(*args, **kwds)

Track time changes of event objects.

This provides a list of changes while the context manager was active and also triggers times_changed signals.

If the code running inside the with block of this context manager raises an exception, no signals will be triggered.

Parameters

• auto_extend – Whether entry parents will get their boundaries automatically extended or not. Passing 'start' will extend only start datetime, 'end' to extend only end datetime.

• user – The User that will trigger time changes.
Indico Documentation, Release 2.3-dev

```python
indico.modules.events.util.update_object_principals(obj, new_principals, read_access=False, full_access=False, permission=None)
```

Updates an object’s ACL with a new list of principals

Exactly one argument out of `read_access`, `full_access` and `role` must be specified.

**Parameters**

- **obj** – The object to update. Must have `acl_entries`
- **new_principals** – The set containing the new principals
- **read_access** – Whether the read access ACL should be updated
- **full_access** – Whether the full access ACL should be updated
- **permission** – The role ACL that should be updated

**Settings**

```python
class indico.modules.events.settings.EventACLProxy(proxy)
```

Bases: `indico.core.settings.proxy.ACLProxyBase`

Proxy class for event-specific ACL settings

- **add_principal** *(event, *args, **kwargs)*
  Adds a principal to an ACL

  **Parameters**

  - **event** – Event (or its ID)
  - **name** – Setting name
  - **principal** – A `User` or a `GroupProxy`

- **contains_user** *(event, *args, **kwargs)*
  Checks if a user is in an ACL.

  To pass this check, the user can either be in the ACL itself or in a group in the ACL.

  **Parameters**

  - **event** – Event (or its ID)
  - **name** – Setting name
  - **user** – A `User`

- **get** *(event, *args, **kwargs)*
  Retrieves an ACL setting

  **Parameters**

  - **event** – Event (or its ID)
  - **name** – Setting name

- **merge_users** *(target, source)*
  Replaces all ACL user entries for `source` with `target`

- **remove_principal** *(event, *args, **kwargs)*
  Removes a principal from an ACL
Parameters

- **event** – Event (or its ID)
- **name** – Setting name
- **principal** – A User or a GroupProxy

```python
set(event, *args, **kwargs)
```

Replaces an ACL with a new one

Parameters

- **event** – Event (or its ID)
- **name** – Setting name
- **acl** – A set containing principals (users/groups)

```python
class indico.modules.events.settings.EventSettingProperty(proxy, name, default=<object object>, attr=None)
```

Bases: indico.core.settings.proxy.SettingsProxy

```python
attr = u'event'
```

```python
class indico.modules.events.settings.EventSettingsProxy(module, defaults=None, strict=True, acls=None, converters=None)
```

Bases: indico.core.settings.proxy.SettingsProxyBase

Proxy class to access event-specific settings for a certain module

```python
acl_proxy_class
```

alias of EventACLProxy

```python
delete(event, *args, **kwargs)
```

Deletes settings.

Parameters

- **event** – Event (or its ID)
- **names** – One or more names of settings to delete

```python
delete_all(event, *args, **kwargs)
```

Deletes all settings.

Parameters **event** – Event (or its ID)

```python
get(event, *args, **kwargs)
```

Retrieves the value of a single setting.

Parameters

- **event** – Event (or its ID)
- **name** – Setting name
- **default** – Default value in case the setting does not exist

Returns The setting’s value or the default value

```python
get_all(event, *args, **kwargs)
```

Retrieves all settings

Parameters
• **event** – Event (or its ID)
• **no_defaults** – Only return existing settings and ignore defaults.

**Returns** Dict containing the settings

**query**
Returns a query object filtering by the proxy’s module.

**set**(event, *args, **kwargs)
Sets a single setting.

**Parameters**

• **event** – Event (or its ID)
• **name** – Setting name
• **value** – Setting value; must be JSON-serializable

**set_multi**(event, *args, **kwargs)
Sets multiple settings at once.

**Parameters**

• **event** – Event (or its ID)
• **items** – Dict containing the new settings

**class** indico.modules.events.settings.ThemeSettingsProxy
**Bases:** object

**defaults**

**get_themes_for**(**kwargs)

**settings**

**themes**

indico.modules.events.settings.event_or_id(f)

### 5.1.2 Abstract

**Todo:** Docstrings (module, models, operations, utilities, settings)

**Models**

**class** indico.modules.events.abstracts.models.abstracts.Abstract(**kwargs)
**Bases:**  indico.modules.events.models.reviews.ProposalMixin, indico.modules.events.models.reviews.ProposalRevisionMixin,  indico.core.db.sqlalchemy.descriptions.DescriptionMixin,  indico.modules.events.contributions.models.contributions.CustomFieldsMixin,  indico.modules.events.models.persons.AuthorsSpeakersMixin,  sqlalchemy.ext.declarative.api.Model

Represents an abstract that can be associated to a Contribution.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```
AUTHORS_SPEAKERS_DISPLAY_ORDER_ATTR = u'display_order_key_lastname'
accepted_contribution
accepted_contribution_id
accepted_track
accepted_track_id
call_for_proposals_attr = u'cfa'
can_access(user)
can_comment(user, check_state=False)
can_convene(user)
can_edit(user)
can_judge(user, check_state=False)
can_review(user, check_state=False)
can_review(user)
can_withdraw(user, check_state=False)
candidate_contribution
candidate_tracks
create_comment_endpoint = u'abstracts.comment_abstract'
create_judgment_endpoint = u'abstracts.judge_abstract'
create_review_endpoint = u'abstracts.review_abstract'
data_by_field
default_render_mode = 2
delete_comment_endpoint = u'abstracts.delete_abstract_comment'
duplicate_of
duplicate_of_id
edit_comment_endpoint = u'abstracts.edit_abstract_comment'
edit_review_endpoint = u'abstracts.edit_review'
edit_track_mode
event
event_id
field_values
    Data stored in abstract/contribution fields
friendly_id
getReviewedForGroups(user, include_reviewed=False)
get_timeline(user=None)
get_track_question_scores()
```
get_track_reviewing_state(track)
get_track_score(track)
id
is_deleted
is_in_final_state
judge
    User who judged the abstract
judge_id
    ID of the user who judged the abstract
judgment_comment
judgment_dt
locator
    Defines a smart locator property.
    This behaves pretty much like a normal read-only property and the decorated function should return a dict
    containing the necessary data to build a URL for the object.
    This decorator should usually be applied to a method named locator as this name is required for
    get_locator to find it automatically when just passing the object.
    If you need more than one locator, you can define it like this:

        @locator_property
        def locator(self):
            return {...}
        @locator.other
        def locator(self):
            return {...}

    The other locator can then be accessed by passing obj.locator.other to the code expecting an
    object with a locator.

marshmallow_aliases = {u'_description': u'content'}
merged_into
merged_into_id
modified_by
modified_by_id
modified_dt
person_links
    Persons associated with this abstract
possible_render_modes = set([<RenderMode.markdown: 2>])
proposal_type = u'abstract'
public_state
render_mode = 2
reset_state()
reviewed_for_tracks
reviewing_state
revisions_enabled = False
score
state
submission_comment
submitted_contrib_type
submitted_contrib_type_id
submitted_dt
submitted_for_tracks
submitter
   User who submitted the abstract
submitter_id
   ID of the user who submitted the abstract
title
user_owns (user)
uuid
verbose_title
class indico.modules.events.abstracts.models.abstracts.AbstractPublicState
   Bases: indico.util.struct.enum.RichIntEnum
      accepted = 3
      awaiting = -1
      duplicate = 6
      invited = 7
      merged = 5
      rejected = 4
      under_review = -2
      withdrawn = 2
class indico.modules.events.abstracts.models.abstracts.AbstractReviewingState
   Bases: indico.util.struct.enum.RichIntEnum
      conflicting = 3
      in_progress = 1
      mixed = 5
      negative = 4
      not_started = 0
      positive = 2
class indico.modules.events.abstracts.models.abstracts.AbstractState
   Bases: indico.util.struct.enum.RichIntEnum
accepted = 3
duplicate = 6
invited = 7
merged = 5
rejected = 4
submitted = 1
withdrawn = 2

class indico.modules.events.abstracts.models.abstracts.EditTrackMode
    Bases: int, indico.util.struct.enum.IndicoEnum
    both = 1
    none = 0
    reviewed_for = 2

class indico.modules.events.abstracts.models.call_for_abstracts.CallForAbstracts(event)
    Bases: object
    Proxy class to facilitate access to the call for abstracts settings
    allow_attachments
    allow_comments
    allow_contributors_in_comments
    allow_convener_judgment
    announcement
    can_edit_abstracts(user)
    can_submit_abstracts(user)
    close()
    end_dt
    has Ended
    has_started
    is_open
    is_scheduled
    judgment_instructions
    modification_end_dt
    modification_ended
    open()
    rating_range
    reviewing_instructions
    schedule(start_dt, end_dt, modification_end_dt)
    start_dt
    submission_instructions
class indico.modules.events.abstracts.models.comments.AbstractComment(**kwargs)
Bases: indico.modules.events.models.reviews.ProposalCommentMixin, indico.core.db.sqlalchemy.review_comments.ReviewCommentMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

abstract
abstract_id
can_edit(user)
can_view(user)
created_dt
id
is_deleted
locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

marshmallow_aliases = {u'_text': u'text'}
modified_by
modified_by_id
modified_dt
render_mode = 2
user
user_backref_name = u'abstract_comments'
user_id
user_modified_backref_name = u'modified_abstract_comments'
visibility
class indico.modules.events.abstracts.models.email_logs.AbstractEmailLogEntry(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

abstract
abstract_id
body
classmethod create_from_email(email_data, email_tpl, user=None)
Create a new log entry from the data used to send an email

Parameters
• email_data – email data as returned from make_email
• email_tpl – the abstract email template that created the email
• user – the user who performed the action causing the notification
data
email_template
email_template_id
id
recipients
sent_dt
subject
user
user_id
class indico.modules.events.abstracts.models.email_templates.AbstractEmailTemplate(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Model

Represents an email template for abstracts notifications.
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
body
    The body of the template
event
event_id
extra_cc_emails
    List of extra email addresses to be added as CC in the email
id
**include_authors**
Whether to include authors’ email addresses as To for emails

**include_coauthors**
Whether to include co-authors’ email addresses as CC for emails

**include_submitter**
Whether to include the submitter’s email address as To for emails

**locator**
Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named ` locator` as this name is required for
`get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `objlocator.other` to the code expecting an object with a locator.

**position**
The relative position of the template in the list of templates

**reply_to_address**
The address to use as Reply-To in the email

**rules**
Conditions need to be met to send the email

**stop_on_match**
Whether to stop checking the rest of the conditions when a match is found

**subject**
The subject of the email

**title**

```python
class indico.modules.events.abstracts.models.fields.AbstractFieldValue(**kwargs)
Bases: indico.modules.events.contributions.models.fields.ContributionFieldValueBase
```

Store a field values related to abstracts.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

**abstract_id**

**contribution_field**
contribution_field_backref_name = u'abstract_values'
contribution_field_id
data
class indico.modules.events.abstracts.models.files.AbstractFile(**kwargs)
    Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    abstract
    abstract_id
    add_file_date_column = False
    content_type
    The MIME type of the file
    created_dt = None
    filename
    The name of the file
    id
    locator
    md5
    An MD5 hash of the file.
    Automatically assigned when save() is called.
    size
    The size of the file (in bytes).
    Automatically assigned when save() is called.
    storage_backend
    storage_file_id
class indico.modules.events.abstracts.models.persons.AbstractPersonLink(*args, **kwargs)
    Bases: indico.modules.events.models.persons.PersonLinkBase
    Association between EventPerson and Abstract.
    abstract_id
    author_type
    display_order
    id
    is_speaker
    locator
    Defines a smart locator property.
This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
object_relationship_name = u'abstract'

person
    person_id
    person_link_backref_name = u'abstract_links'
    person_link_unique_columns = (u'abstract_id',)

class indico.modules.events.abstracts.models.review_questions.AbstractReviewQuestion(**kwargs):
    Bases: indico.core.db.sqlalchemy.review_questions.ReviewQuestionMixin,
          sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in kwargs.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    description
    event
        event_backref_name = u'abstract_review_questions'
        event_id
        field
        field_data
        field_type
        id
        is_deleted
        is_required
        locator
            Defines a smart locator property.

            This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
```
This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def other(locator=self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
position
title
```

```python
class indico.modules.events.abstracts.models.review_ratings.AbstractReviewRating(**kwargs)
    Bases: indico.core.db.sqlalchemy.review_ratings.ReviewRatingMixin, sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in `kwargs`.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

id
question
question_class
    alias of indico.modules.events.abstracts.models.review_questions.AbstractReviewQuestion
question_id
review
review_class
    alias of indico.modules.events.abstracts.models.reviews.AbstractReview
review_id
value
```

```python
class indico.modules.events.abstracts.models.reviews.AbstractAction
    Bases: indico.util.struct.enum.RichIntEnum
    accept = 1
    change_tracks = 3
    mark_as_duplicate = 4
    merge = 5
    reject = 2
```

```python
class indico.modules.events.abstracts.models.reviews.AbstractCommentVisibility
    Bases: indico.util.struct.enum.RichIntEnum
```
Most to least restrictive visibility for abstract comments

```
contributors = 4
conveners = 2
judges = 1
reviewers = 3
users = 5
```

```python
class indico.modules.events.abstracts.models.reviews.AbstractReview(**kwargs)
Bases: indico.modules.events.models.reviews.ProposalReviewMixin, indico.core.db.sqlalchemy.descriptions.RenderModeMixin, sqlalchemy.ext.declarative.api.Model

Represents an abstract review, emitted by a reviewer

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```
abstract
abstract_id

can_edit (user, check_state=False)

can_view (user)

comment

created_dt

default_render_mode = 2

group_attr = u'track'

id

locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

```
marshmallow_aliases = {u'_comment': u'comment'}
```
modified_dt
possible_render_modes = set([<RenderMode.markdown: 2>])
proposed_action
proposed_contribution_type
proposed_contribution_type_id
proposed_related_abstract
proposed_related_abstract_id
proposed_tracks
render_mode = 2
revision_attr = u'abstract'
score
track
track_id
user
user_id
visibility

Operations

indico.modules.events.abstracts.operations.add_abstract_files(abstract, files, log_action=True)

indico.modules.events.abstracts.operations.close_cfa(event)

indico.modules.events.abstracts.operations.create_abstract(event, abstract_data, custom_fields_data=None, send_notifications=False, submitter=None, is_invited=False)

indico.modules.events.abstracts.operations.create_abstract_comment(abstract, comment_data)

indico.modules.events.abstracts.operations.create_abstract_review(abstract, track, user, review_data, questions_data)

indico.modules.events.abstracts.operations.delete_abstract(abstract, delete_contrib=False)

indico.modules.events.abstracts.operations.delete_abstract_comment(comment)

indico.modules.events.abstracts.operations.delete_abstract_files(abstract, files)
indico.modules.events.abstracts.operations.**judge_abstract**(abstract, abstract_data, judgment, judge, contrib_session=None, merge_persons=False, send_notifications=False)

indico.modules.events.abstracts.operations.**open_cfa**(event)

indico.modules.events.abstracts.operations.**reset_abstract_state**(abstract)

indico.modules.events.abstracts.operations.**schedule_cfa**(event, start_dt, end_dt, modification_end_dt)

indico.modules.events.abstracts.operations.**update_abstract**(abstract, abstract_data, custom_fields_data=None)

indico.modules.events.abstracts.operations.**update_abstract_comment**(comment, comment_data)

indico.modules.events.abstracts.operations.**update_abstract_review**(review, review_data, questions_data)

indico.modules.events.abstracts.operations.**update_reviewed_for_tracks**(abstract, tracks)

indico.modules.events.abstracts.operations.**withdraw_abstract**(abstract)

**Utilities**

indico.modules.events.abstracts.util.**build_default_email_template**(event, tpl_type)

Build a default e-mail template based on a notification type provided by the user.

indico.modules.events.abstracts.util.**can_create_invited_abstracts**(event)

indico.modules.events.abstracts.util.**clear_boa_cache**(event)

Delete the cached book of abstract

indico.modules.events.abstracts.util.**create_boa**(event)

Create the book of abstracts if necessary

**Returns** The path to the PDF file

indico.modules.events.abstracts.util.**create_boa_tex**(event)

Create the book of abstracts as a LaTeX archive.

**Returns** A BytesIO containing the zip file.

indico.modules.events.abstracts.util.**create_mock_abstract**(args, **kwargs)

Create a mock abstract that can be used in previews.

Brace for geek references.

indico.modules.events.abstracts.util.**filter_field_values**(fields, can_manage, owns_abstract)
Generates a spreadsheet data from a given abstract list.

**Parameters**

- `abstracts` – The list of abstracts to include in the file
- `static_item_ids` – The abstract properties to be used as columns
- `dynamic_items` – Contribution fields as extra columns

Return a dict of event ids and the abstract submission related roles the user has in that event.

**Parameters**

- `user` – A User
- `dt` – Only include events taking place on/after that date

Return a dict of event ids and the abstract reviewing related roles the user has in that event.

**Parameters**

- `user` – A User
- `dt` – Only include events taking place on/after that date

Get the numbers of abstracts per track for a specific user.

Note that this does not take into account if the user is a reviewer for a track; it just checks whether the user has reviewed an abstract in a track or not.

**Returns** A dict mapping tracks to dicts containing the counts.

Get the list of abstracts where the user is a reviewer/convener

Get the list of tracks where the user is a reviewer/convener

Get the list of visible reviewed tracks for the user.

Extends the abstract WTForm to add the extra fields.

Each extra field will use a field named `custom_ID`.

**Parameters**

- `event` – The Event for which to create the abstract form.
- `user` – The user who is going to use the form.
• **notification_option** – Whether to add a field to the form to disable triggering notifications for the abstract submission.

• **management** – Whether the form is used in the management area

• **invited** – Whether the form is used to create an invited abstract

**Returns** An AbstractForm subclass.

**Placeholders**

```python
class indico.modules.events.abstracts.placeholders.EventTitlePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The title of the event'
    name = u'event_title'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.EventURLPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The URL of the event'
    name = u'event_url'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.AbstractIDPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The ID of the abstract'
    name = u'abstract_id'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.AbstractTitlePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The title of the abstract'
    name = u'abstract_title'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.AbstractURLPlaceholder
    Bases: indico.util.placeholders.Placeholder
    advanced = True
    description = lu'The direct URL of the abstract'
    name = u'abstract_url'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.AbstractInvitationURLPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The link to submit an invited abstract'
    name = u'invitation_url'
    classmethod render(abstract)
```
class indico.modules.events.abstracts.placeholders.AbstractTrackPlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = lu'The name of the destination track'
    name = u'abstract_track'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.AbstractSessionPlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = lu'The name of the destination session'
    name = u'abstract_session'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.PrimaryAuthorsPlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = lu'The names of the primary authors (separated by commas)'
    name = u'primary_authors'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.CoAuthorsPlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = lu'The names of the co-authors (separated by commas)'
    name = u'co_authors'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.SubmitterNamePlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = lu'The full name of the submitter, no title'
    name = u'submitter_name'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.SubmitterFirstNamePlaceholder
    Bases: indico.util.placeholders.Placeholder

    advanced = True
    description = lu'The first name of the submitter'
    name = u'submitter_first_name'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.SubmitterLastNamePlaceholder
    Bases: indico.util.placeholders.Placeholder

    advanced = True
    description = lu'The last name of the submitter'
    name = u'submitter_last_name'
    classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.SubmitterTitlePlaceholder
    Bases: indico.util.placeholders.Placeholder
description = lu'The title of the submitter (Dr, Prof., etc...)'
name = u'submitter_title'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.TargetAbstractIDPlaceholder
Bases: indico.util.placeholders.Placeholder
description = lu'The ID of the target abstract (merge or duplicate)'
name = u'target_abstract_id'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.TargetAbstractTitlePlaceholder
Bases: indico.util.placeholders.Placeholder
description = lu'The title of the target abstract (merge or duplicate)'
name = u'target_abstract_title'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.TargetSubmitterNamePlaceholder
Bases: indico.util.placeholders.Placeholder
advanced = True
description = lu"The full name of the target abstract's submitter, no title (merge or duplicate)"
name = u'target_submitter_name'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.TargetSubmitterFirstNamePlaceholder
Bases: indico.util.placeholders.Placeholder
advanced = True
description = lu"The first name of the target abstract's submitter (merge or duplicate)"
name = u'target_submitter_first_name'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.TargetSubmitterLastNamePlaceholder
Bases: indico.util.placeholders.Placeholder
advanced = True
description = lu"The last name of the target abstract's submitter (merge or duplicate)"
name = u'target_submitter_last_name'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.JudgmentCommentPlaceholder
Bases: indico.util.placeholders.Placeholder
description = lu'Comments written by event organizer (upon final decision)'
name = u'judgment_comment'
classmethod render(abstract)
class indico.modules.events.abstracts.placeholders.ContributionTypePlaceholder
Bases: indico.util.placeholders.Placeholder

5.1. API reference
description = lu'The contribution type that is associated to the abstract'
name = u'contribution_type'
classmethod render(abstract)

class indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder
    Bases: indico.util.placeholders.Placeholder
    advanced = True
description = lu'Contribution URL'
name = u'contribution_url'
classmethod render(abstract)

Settings

class indico.modules.events.abstracts.settings.BOACorrespondingAuthorType
    Bases: indico.util.struct.enum.RichEnum
    none = u'none'
speakers = u'speakers'
s submitter = u'submitter'

class indico.modules.events.abstracts.settings.BOALinkFormat
    Bases: indico.util.struct.enum.RichEnum
    LaTeX book of abstracts link format setting
    value is a 2-tuple of strings: first is the hyperref option to use second sets additional tex commands
    colorlinks = (u'[colorlinks]', u'')
    frame = (u'', u'')
    unstyled = (u'[hidelinks]', u'')

class indico.modules.events.abstracts.settings.BOASortField
    Bases: indico.util.struct.enum.RichEnum
    abstract_title = u'title'
    board_number = u'board_number'
    id = u'id'
    schedule = u'schedule'
    schedule_board_number = u'schedule_board_number'
    session_board_number = u'session_board_number'
    session_schedule_board = u'session_schedule_board'
    session_title = u'session_title'
    speaker = u'speaker'
5.1.3 Agreement

**Todo:** Docstrings (module, models, utilities)

### Models

```python
class indico.modules.events.agreements.models.agreements.Agreement(**kwargs):
    Bases: sqlalchemy.ext.declarative.api.Model

    Agreements between a person and Indico

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in `kwargs`.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    `accept(from_ip, reason=None, on_behalf=False)`

    `accepted`

    `attachment`

    Attachment

    `attachment_filename`

    Filename and extension of the attachment

    `belongs_to(person)`

    `static create_from_data(event, type_, person)`

    `data`

    Definition-specific data of the agreement

    `definition`

    `event`

    The Event this agreement is associated with

    `event_id`

    ID of the event

    `id`

    Entry ID

    `identifier`

    Unique identifier within the event and type

    `is_orphan()`

    `locator`

    `pending`

    `person_email`

    Email of the person agreeing

    `person_name`

    Full name of the person agreeing
```
**reason**
- Explanation as to why the agreement was accepted/rejected

**reject** (*from_ip*, *reason=None*, *on_behalf=False*)
- rejected

**render** (*form, **kwargs*)
- reset()

**signed_dt**
- The date and time the agreement was signed

**signed_from_ip**
- The IP from which the agreement was signed

**signed_on_behalf**

**state**
- A AgreementState

**timestamp**
- The date and time the agreement was created

**type**
- Type of agreement

**user**
- The user this agreement is linked to

**user_id**
- ID of a linked user

**uuid**
- Entry universally unique ID

```python
class indico.modules.events.agreements.models.agreements.AgreementState
   Bases: indico.util.struct.enum.RichIntEnum
   accepted = 1
   accepted_on_behalf = 3
      agreement accepted on behalf of the person
   pending = 0
   rejected = 2
   rejected_on_behalf = 4
      agreement rejected on behalf of the person
```

**Utilities**

```python
indico.modules.events.agreements.util.get_agreement_definitions()
indico.modules.events.agreements.util.send_new_agreements(event, name, people, email_body, cc_addresses, from_address)
```

Creates and send agreements for a list of people on a given event.

**Parameters**
- **event** – The Event associated with the agreement
- **name** – The agreement type matching an AgreementDefinition name
- **people** – The list of people for whom agreements will be created
- **email_body** – The body of the email
- **cc_addresses** – Email addresses to send CCs to
- **from_address** – Email address of the sender

### Placeholders

```python
class indico.modules.events.agreements.placeholders.AgreementLinkPlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = u'Link to the agreement page'
    name = u'agreement_link'

    @classmethod
    def render(cls, definition, agreement)
        required = True
```

```python
class indico.modules.events.agreements.placeholders.PersonNamePlaceholder
    Bases: indico.util.placeholders.Placeholder

    description = u'Name of the person'
    name = u'person_name'

    @classmethod
    def render(cls, definition, agreement)
```

### 5.1.4 Contribution

**Todo:** Docstrings (module, models, operations, utilities)

### Models

```python
class indico.modules.events.contributions.models.contributions.Contribution(**kwargs)

    ATTACHMENT_FOLDER_ID_COLUMN = u'contribution_id'

    PRELOAD_EVENT_ATTACHED_ITEMS = True

    PRELOAD_EVENT_NOTES = True

    abstract
    abstract_id
```

5.1. API reference
access_key = None

acl_entries

classmethod allocate_friendly_ids(event, n)
    Allocate n Contribution friendly_ids.
    This is needed so that we can allocate all IDs in one go. Not doing so could result in DB deadlocks. All
    operations that create more than one contribution should use this method.

    Parameters
    • event – the Event in question
    • n – the number of ids to pre-allocate

allow_relationship_preloading = True

board_number

can_manage(user, permission=None, allow_admin=True, check_parent=True, explicit_permission=False)

code
default_render_mode = 2
disallowed_protection_modes = frozenset([])
duration
duration_display
    The displayed duration of the contribution.
    This is the duration of the poster session if applicable, otherwise the duration of the contribution itself.
duration_poster
date
date_display
    The displayed end time of the contribution.
    This is the end time of the poster session if applicable, otherwise the end time of the contribution itself.
date_poster
event
event_id

field_values
    Data stored in abstract/contribution fields

friendly_id
    The human-friendly ID for the contribution

get_non_inheriting_objects()
    Get a set of child objects that do not inherit protection.
id

inherit_location

inheriting_have_acl = True

is_deleted

is_paper_reviewer(user)
is_scheduled

is_user_associated(user, check_abstract=False)

keywords

location_backref_name = u'contributions'

location_parent

locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}
@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

own_address

own_no_access_contact = None

own_room

own_room_id

own_room_name

own_venue

own_venue_id

own_venue_name

paper

paper_content_reviewers

Paper content reviewers

paper_judges

Paper reviewing judges

paper_layout_reviewers

Paper layout reviewers

pending_paper_files

Paper files not submitted for reviewing

person_links

Persons associated with this contribution

possible_render_modes = set([<RenderMode.html: 1>, <RenderMode.markdown: 2>])
classmethod preload_acl_entries(event)
protection_mode
protection_parent
references
    External references associated with this contribution
render_mode
session
session_block
session_block_id
session_id
start_dt
start_dt_display
    The displayed start time of the contribution.
    This is the start time of the poster session if applicable, otherwise the start time of the contribution itself.
start_dt_poster
subcontribution_count
subcontributions
submitters
timetable_entry
title
track
track_id
type
type_id
verbose_title

class indico.modules.events.contributions.models.contributions.CustomFieldsMixin
    Bases: object

    Methods to process custom field data.
    get_field_value(field_id, raw=False)
    set_custom_field(field_id, field_value)

class indico.modules.events.contributions.models.fields.ContributionField(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
description
event
event_id
field
field_data
field_type
filter_choices
id
is_active
is_public
is_required
is_user_editable
legacy_id
locator

Defines a smart locator property.
This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

mgmt_field
position
title
visibility

class ContributionFieldValue(**kwargs):
    contribution_field

A simple constructor that allows initialization from `kwargs`.
Sets attributes on the constructed instance using the names and values in `kwargs`.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

class ContributionFieldValueBase

contribution_field_backref_name = u'contribution_values'
class ContributionFieldValueBase(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    contribution_field = <RelationshipProperty at 0x7fcf712f16b0; no key>
    contribution_field_backref_name = None
    The name of the backref on the ContributionField

data = Column(None, JSONB(astext_type=Text()), table=None, nullable=False)

class ContributionFieldVisibility
    Bases: indico.util.struct.enum.RichIntEnum
    managers_and_submitters = 2
    managers_only = 3
    public = 1

class ContributionPersonLink(*args, **kwargs)
    Bases: indico.modules.events.models.persons.PersonLinkBase
    Association between EventPerson and Contribution.
    author_type
    contribution_id
    display_order
    id
    is_author
    is_speaker
    is_submitter
locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def other(locator=self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

```python
object_relationship_name = u'contribution'

person

person_id

person_link_backref_name = u'contribution_links'

person_link_unique_columns = (u'contribution_id',)

class indico.modules.events.contributions.models.persons.SubContributionPersonLink(*args, **kwargs)

    Bases: indico.modules.events.models.persons.PersonLinkBase

    Association between EventPerson and SubContribution.

    author_type = 0

    display_order

    id

    is_speaker = True

object_relationship_name = u'subcontribution'

person

person_id

person_link_backref_name = u'subcontribution_links'

person_link_unique_columns = (u'subcontribution_id',)

subcontribution_id

class indico.modules.events.contributions.models.principals.ContributionPrincipal(**kwargs)

    Bases: indico.core.db.sqlalchemy.principals.PrincipalPermissionsMixin,
           sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in kwargs.
```
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.

```python
allow_category_roles = True
allow_emails = True
allow_event_roles = True
category_role
category_role_id
contribution_id
    The ID of the associated contribution
disallowed_protection_modes = frozenset([])
email
event_role
event_role_id
full_access
id
    The ID of the acl entry
ip_network_group = None
ip_network_group_id = None
local_group
local_group_id
multipass_group_name
multipass_group_provider
permissions
principal_backref_name = u'in_contribution_acls'
principal_for = u'Contribution'
read_access
type
unique_columns = (u'contribution_id',)
user
user_id
```

```python
class ContributionReference(**kwargs)
    Bases: indico.modules.events.models.references.ReferenceModelBase
```

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.

```python
contribution_id
id
```
reference_backref_name = u'contribution_references'
reference_type
reference_type_id
value

```python
class indico.modules.events.contributions.models.references.SubContributionReference(**kwargs):
    reference_backref_name = u'subcontribution_references'
    reference_type
    reference_type_id
    subcontribution_id
    value
```

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in `kwargs`.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
id
reference_backref_name = u'subcontribution_references'
reference_type
reference_type_id
subcontribution_id
value
```

```python
class indico.modules.events.contributions.models.subcontributions.SubContribution(**kwargs):
    ATTACHMENT_FOLDER_ID_COLUMN = u'subcontribution_id'
    PRELOAD_EVENT_ATTACHED_ITEMS = True
    PRELOAD_EVENT_NOTES = True
    can_access(user, **kwargs)
    can_manage(user, permission=None, **kwargs)
    code
    contribution_id
    default_render_mode = 2
    duration
    event
    friendly_id
        The human-friendly ID for the sub-contribution
    get_access_list()
    get_manager_list(recursive=False, include_groups=True)
    id
    is_deleted
    is_protected
    location_parent
```
**locator**

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

**person_links**

Persons associated with this contribution

**position**

```python
possible_render_modes = set([<RenderMode.html: 1>, <RenderMode.markdown: 2>])
```

**references**

External references associated with this contribution

**render_mode**

**session**

Convenience property so all event entities have it

**speakers**

**timetable_entry**

Convenience property so all event entities have it

**title**

```python
class ContributionType(**kwars)
Bases: sqlalchemy.ext.declarative.api.Model
```

A simple constructor that allows initialization from `kwars`.

Sets attributes on the constructed instance using the names and values in `kwars`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

**description**

**event**

**event_id**

**id**

**is_private**
locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

name

Operations

```python
indico.modules.events.contributions.operations.create_contribution(event, contrib_data, custom_fields_data=None, session_block=None, extend_parent=False)
```

```python
indico.modules.events.contributions.operations.create_contribution_from_abstract(*args, **kwargs)
```

```python
indico.modules.events.contributions.operations.create_subcontribution(contrib, data)
```

```python
indico.modules.events.contributions.operations.delete_contribution(contrib)
```

```python
indico.modules.events.contributions.operations.delete_subcontribution(subcontrib)
```

```python
indico.modules.events.contributions.operations.update_contribution(*args, **kwargs)
```

Update a contribution

**Parameters**

- `contrib` – The Contribution to update
- `contrib_data` – A dict containing the data to update
- `custom_fields_data` – A dict containing the data for custom fields.

**Returns** A dictionary containing information related to the update. `unscheduled` will be true if the modification resulted in the contribution being unscheduled. In this case `undo_unschedule` contains the necessary data to re-schedule it (undoing the session change causing it to be unscheduled)
Utilities

indico.modules.events.contributions.util.contribution_type_row(contrib_type)

Return a tuple consisting of spreadsheet columns and respective contribution values

indico.modules.events.contributions.util.generate_spreadsheet_from_contributions(contributions)

Return a tuple consisting of spreadsheet columns and respective contribution values

indico.modules.events.contributions.util.get_boa_export_formats()

indico.modules.events.contributions.util.get_contribution_ical_file(contrib)

Get a list of contributions in which the user has submission rights

indico.modules.events.contributions.util.get_events_with_linked_contributions(user, dt=None)

Returns a dict with keys representing event_id and the values containing data about the user rights for contributions within the event

Parameters

• user – A User
• dt – Only include events taking place on/after that date

indico.modules.events.contributions.util.has_contributions_with_user_as_submitter(event, user)

import_contributions_from_csv(event, f)

Import timetable contributions from a CSV file into an event.

indico.modules.events.contributions.util.make_contribution_form(event)

Extends the contribution WTForm to add the extra fields.

Each extra field will use a field named custom_ID.

Parameters

• event – The Event for which to create the contribution form.

Returns

A ContributionForm subclass.

indico.modules.events.contributions.util.render_archive(event, contribs, sort_by, cls)

indico.modules.events.contributions.util.render_pdf(event, contribs, sort_by, cls)

indico.modules.events.contributions.util.serialize_contribution_for_ical(contrib)

indico.modules.events.contributions.util.serialize_contribution_person_link(person_link, is_submitter=None)

Serialize ContributionPersonLink to JSON-like object

indico.modules.events.contributions.util.sort_contribs(contribs, sort_by)

5.1.5 Feature

Todo: Docstrings (module, utilities)
Utilities

indico.modules.events.features.util.format_feature_names(names)

Get a set containing the names of features which are not available for an event.

indico.modules.events.features.util.get_enabled_features(event, only_explicit=False)

Returns a set of enabled feature names for an event.

indico.modules.events.features.util.get_feature_definition(name)

Gets a feature definition.

indico.modules.events.features.util.get_feature_definitions()

Gets a dict containing all feature definitions.

indico.modules.events.features.util.is_feature_enabled(event, name)

Checks if a feature is enabled for an event.

Parameters

• event – The event (or event ID) to check.
• name – The name of the feature.

indico.modules.events.features.util.require_feature(event, name)

Raises a NotFound error if a feature is not enabled.

Parameters

• event – The event (or event ID) to check.
• name – The name of the feature.

indico.modules.events.features.util.set_feature_enabled(event, name, state)

Enables/disables a feature for an event.

Parameters

• event – The event.
• name – The name of the feature.
• state – If the feature is enabled or not.

Returns Boolean indicating if the state of the feature changed.

5.1.6 Layout

Todo: Docstrings (module, models, utilities)

Models

class indico.modules.events.layout.models.images.ImageFile(**kwargs)

Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
content_type
    The MIME type of the file

created_dt
    The date/time when the file was uploaded

event
    The event the image belongs to

filename
    The name of the file

id
    The ID of the file

locator

md5
    An MD5 hash of the file.
    Automatically assigned when `save()` is called.

size
    The size of the file (in bytes).
    Automatically assigned when `save()` is called.

storage_backend

storage_file_id

version_of = None
```

```python
class indico.modules.events.layout.models.menu.EventPage(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from `kwargs`.

    Sets attributes on the constructed instance using the names and values in `kwargs`.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    event
        The Event which contains the page

    event_id
        The ID of the event which contains the page

    html
        The rendered HTML of the page

    id
        The ID of the page

    is_default

    locator
```
class indico.modules.events.layout.models.menu.MenuEntry(**kwargs)
Bases: indico.modules.events.layout.models.menu.MenuEntryMixin, sqlalchemy.ext.declarative.api.Model

children
    The children menu entries and parent backref

event
    The Event containing the menu entry

event_id
    The ID of the event which contains the menu

static get_for_event(event)

id
    The ID of the menu entry

insert(parent, position)

is_enabled
    Whether the entry is visible in the event’s menu

is_root

link_url
    The target URL of a custom link

move(to)

name
    The name of the menu entry (to uniquely identify a default entry for a given event)

new_tab
    Whether the menu entry should be opened in a new tab or window

page
    The page of the menu entry

page_id
    The page ID if the entry is a page

parent_id
    The ID of the parent menu entry (NULL if root menu entry)

plugin
    The name of the plugin from which the entry comes from (NULL if the entry does not come from a plugin)

position
    The relative position of the entry in the menu

title
    The title of the menu entry (to be displayed to the user)

type
    The type of the menu entry

class indico.modules.events.layout.models.menu.MenuEntryMixin(**kwargs)
Bases: object

default_data

event_ref

is_internal_link
is_link
is_orphaned
is_page
is_plugin_link
is_separator
is_user_link
is_visible
localized_title
locator
url

class indico.modules.events.layout.models.menu.MenuEntryType
Bases: indico.util.struct.enum.RichIntEnum

internal_link = 2
page = 5
plugin_link = 4
separator = 1
user_link = 3

class indico.modules.events.layout.models.menu.TransientMenuEntry(event,
is_enabled, name, position, children)
Bases: indico.modules.events.layout.models.menu.MenuEntryMixin

Utilities

class indico.modules.events.layout.util.MenuEntryData(title, name, endpoint=None, position=-1,
is_enabled=True, visible=None, parent=None, static_site=False)
Bases: object

Container to transmit menu entry-related data via signals

The data contained is transmitted via the sidemenu signal and used to build the side menu of an event.

Parameters

• title – str – The title of the menu, displayed to the user. The title should be translated using the normal gettext function, i.e. _('...'), or the plugin’s bound gettext function.

• name – str – Name used to refer to the entry internally. This is never shown to the user. The name must be unique, names from plugins are automatically prefixed with the plugin name and a colon and therefore have to be unique only within the plugin. To mark the entry as active, its name must be specified in the menu_entry_name class attribute of the WP class. For plugins, the plugin name must be specified via the menu_entry_plugin attribute as well.
- **endpoint** – str – The endpoint the entry will point to.

- **position** – int – The desired position of the menu entry. The position is indicative only, relative to the other entries and not the exact position. Entries with the same position will be sorted alphanumerically on their name. A position of `-1` will append the entry at the end of the menu.

- **is_enabled** – bool – Whether the entry should be enabled by default (Default: `True`).

- **visible** – function – Determines if the entry should be visible. This is a simple function which takes only the `event` as parameter and returns a boolean to indicate if the entry is visible or not. It is called whenever the menu is displayed, so the current state of the event/user can be taken into account.

- **parent** – str – The name of the parent entry (None for root entries).

- **static_site** – bool or str – If True, this menu item should be shown in the menu of a static site. When set to a string, the string will be used instead of a mangled version of the endpoint’s URL.

```python
def build_menu_entry_name(name, plugin=None):
    # Builds the proper name for a menu entry.
    # Given a menu entry’s name and optionally a plugin, returns the correct name of the menu entry.
    # Parameters
    # • name – str – The name of the menu entry.
    # • plugin – IndicoPlugin or str – The plugin (or the name of the plugin) which created the entry.
```

```python
def get_css_file_data(event):
    # Builds the URL of a CSS resource.
    # Parameters
    # • event – The Event to get the CSS url for
    # • force_theme – The ID of the theme to override the custom CSS resource only if it exists
    # • for_preview – Whether the URL is used in the CSS preview page
    # Returns The URL to the CSS resource
```

```python
def get_menu_entry_by_name(*args, **kwargs):
    # Check whether the MenuEntry is enabled
```

```python
def menu_entries_for_event(*args, **kwargs):
```

---

5.1. API reference

161
5.1.7 Log

Todo: Docstrings (module, models, utilities)

Models

class indico.modules.events.logs.models.entries.EventLogEntry(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

Log entries for events
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

data
    Type-specific data

event
    The Event this log entry is associated with

event_id
    The ID of the event

id
    The ID of the log entry

kind
    The general kind of operation that was performed

logged_date

logged_dt
    The date/time when the reminder was created

module
    The module the operation was related to (does not need to match something in indico.modules and should be human-friendly but not translated).

realm
    The general area of the event the entry comes from

render()
    Renders the log entry to be displayed.
    If the renderer is not available anymore, e.g. because of a disabled plugin, None is returned.

renderer

summary
    A short one-line description of the logged action. Should not be translated!

type
    The type of the log entry. This needs to match the name of a log renderer.

user
    The user associated with the log entry
user_id
The ID of the user associated with the entry

class indico.modules.events.logs.models.entries.EventLogKind
Bases: int, indico.util.struct.enum.IndicoEnum
    change = 3
    negative = 4
    other = 1
    positive = 2

class indico.modules.events.logs.models.entries.EventLogRealm
Bases: indico.util.struct.enum.RichIntEnum
    emails = 5
    event = 1
    management = 2
    participants = 3
    reviewing = 4

Utilities

indico.modules.events.logs.util.get_log_renderers()

indico.modules.events.logs.util.make_diff_log(changes, fields)
Create a value for log data containing change information.

Parameters

• changes – a dict mapping attributes to (old, new) tuples
• fields – a dict mapping attributes to field metadata. for simple cases this may be a string
  with the human-friendly title, for more advanced fields it should be a dict containing title, a
  type string and a convert callback which will be invoked with a tuple containing the
  old and new value

indico.modules.events.logs.util.render_changes(a, b, type_)
Render the comparison of a and b as HTML.

Parameters

• a – old value
• b – new value
• type – the type determining how the values should be compared

indico.modules.events.logs.util.serialize_log_entry(entry)

class indico.modules.events.logs.renderers.EmailRenderer
Bases: indico.modules.events.logs.renderers.EventLogRendererBase
    name = u'email'
    template_name = u'events/logs/entry_email.html'
class indico.modules.events.logs.renderers.EventLogRendererBase
Bases: object

Base class for event log renderers.

classmethod get_data(entry)
    Returns the entry data in a format suitable for the template.
    This method may be overridden if the entry's data needs to be preprocessed before being passed to the
    template.
    It MUST NOT modify entry.data directly.

name = None
    unique name of the log renderer (matches EventLogEntry.type)

plugin = None
    plugin containing this renderer - assigned automatically

classmethod render_entry(entry)
    Renders the log entry row

    Parameters entry – A EventLogEntry

template_kwargs = {}
    extra kwargs passed to render_template

template_name = None
    template used to render the log entry

class indico.modules.events.logs.renderers.SimpleRenderer
Bases: indico.modules.events.logs.renderers.EventLogRendererBase

classmethod get_data(entry)
    name = u'simple'

    template_kwargs = {u'cmpare': <function render_changes>}

    template_name = u'events/logs/entry_simple.html'

5.1.8 Event Management

class indico.modules.events.management.controllers.RHManageEventBase
Bases: indico.modules.events.controllers.base.RHEventBase, indico.modules.
    events.management.controllers.base.ManageEventMixin

Base class for event management RHs

class indico.modules.events.management.views.WPEventManagement(rh, event_, active_menu_item=None, **kwargs)
    Bases: indico.web.views.WPJinjaMixin, indico.web.views.WPDecorated

Base class for event management pages.

When using this class the template will always have event available; it is not necessary to pass it as a kwarg
when calling the render_template classmethod.

When using the class directly, pass the menu item as a posarg:

    return WPEventManagement.render_template('foobar.html', self.event, 'foobar',
                                             foo='bar')
When subclassing you can set `sidemenu_option` on the class, allowing you to omit it. This is recommended if you have many pages using the same menu item or if you already need to subclass for some other reason (e.g. to set a `template_prefix` or include additional JS/CSS bundles):

```
return WPSomething.render_template('foobar.html', self.event, foo='bar')
```

### 5.1.9 Note

**Todo:** Docstrings (module, models, utilities)

#### Models

```python
class indico.modules.events.notes.models.notes.EventNote(**kwargs)
    Bases: indico.core.db.sqlalchemy.links.LinkMixin, sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    allowed_link_types = frozenset([LinkType.event: 2, LinkType.contribution: 3, LinkType.subcontribution: 4, LinkType.session: 5])
    category = None
    category_id = None
    contribution
    contribution_id
    create_revision (render_mode, source, user)
        Creates a new revision if needed and marks it as undeleted if it was
        Any change to the render mode or the source causes a new revision to be created. The user is not taken into account since a user “modifying” a note without changing things is not really a change.

    current_revision
        The currently active revision of the note

    current_revision_id
        The ID of the current revision

    delete (user)
        Marks the note as deleted and adds a new empty revision

    event
    event_id

    events_backref_name = u'all_notes'

    classmethod get_for_linked_object (linked_object, preload_event=True)
        Gets the note for the given object.
        This only returns a note that hasn’t been deleted.
```
Parameters

- **linked_object** – An event, session, contribution or subcontribution.
- **preload_event** – If all notes for the same event should be pre-loaded and cached in the app context.

```python
classmethod get_or_create(linked_object)
```

Gets the note for the given object or creates a new one.

If there is an existing note for the object, it will be returned even. Otherwise a new note is created.

```python
html
```

The rendered HTML of the note

```python
id
```

The ID of the note

```python
is_deleted
```

If the note has been deleted

```python
link_backref_name = u'note'
```

```python
link_type
```

```python
linked_event
```

```python
linked_event_id
```

```python
locator
```

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
revisions
```

The list of all revisions for the note

```python
session
```

```python
session_block = None
```

```python
session_block_id = None
```

```python
session_id
```

```python
subcontribution
```

```python
subcontribution_id
```
unique_links = True

class indico.modules.events.notes.models.notes.EventNoteRevision(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    created_dt
        The date/time when the revision was created

    html
        The rendered HTML of the note

    id
        The ID of the revision

    note_id
        The ID of the associated note

    render_mode
        How the note is rendered

    source
        The raw source of the note as provided by the user

    user
        The user who created the revision

    user_id
        The user who created the revision

Utilities

    indico.modules.events.notes.util.build_note_api_data(note)
    indico.modules.events.notes.util.build_note_legacy_api_data(note)
    indico.modules.events.notes.util.can_edit_note(obj, user)
        Checks if a user can edit the object’s note
    indico.modules.events.notes.util.get_scheduled_notes(event)
        Gets all notes of scheduled items inside an event

5.1.10 Paper

Todo: Docstrings (module, models, operations, utilities, settings)

The papers module handles the Indico’s Paper Reviewing workflow. The “inputs” of this module are the conference papers, which will be uploaded by the corresponding authors/submitters.
Models

class indico.modules.events.papers.models.call_for_papers.CallForPapers(event)
    Bases: object

    Proxy class to facilitate access to the call for papers settings
    
    announcement
    assignees
    can_access_judging_area(user)
    can_access_reviewing_area(user)
    close()
    content_review_questions
    content_reviewer_deadline
    content_reviewers
    content_reviewing_enabled
    end_dt
    get_questions_for_review_type(review_type)
    get_reviewing_state(reviewing_type)
    has_ended
    has_started
    is_judge(user)
    is_manager(user)
    is_open
    is_reviewer(user, role=None)
    is_staff(user)
    judge_deadline
    judges
    layout_review_questions
    layout_reviewer_deadline
    layout_reviewers
    layout_reviewing_enabled
    managers
    open()
    rating_range
    schedule(start_dt, end_dt)
    set_reviewing_state(reviewing_type, enable)
    start_dt
**user_competences**

class indico.modules.events.papers.models.comments.PaperReviewComment(**kwargs):
    Bases: indico.modules.events.models.reviews.ProposalCommentMixin, indico.core.db.sqlalchemy.review_comments.ReviewCommentMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

`can_edit(user)`

`can_view(user)`

`created_dt`

`id`

`is_deleted`

`locator`

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

`modified_by`

`modified_by_id`

`modified_dt`

`paper_revision`

`render_mode = 2`

`revision_id`

`user`

`user_backref_name = u'review_comments'`

`user_id`

`user_modified_backref_name = u'modified_review_comments'`

`visibility`
class indico.modules.events.papers.models.competences.PaperCompetence(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    competences
    event
    event_id
    id

classmethod merge_users(target, source)

    user
    user_id

class indico.modules.events.papers.models.files.PaperFile(*args, **kwargs)
    Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model

    add_file_date_column = False
    content_type
        The MIME type of the file
    created_dt = None
    filename
        The name of the file
    id

    locator
        Defines a smart locator property.
        This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
        This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.
        If you need more than one locator, you can define it like this:

        @locator_property
        def locator(self):
            return {...}

        @locator.other
        def locator(self):
            return {...}

        The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

    md5
        An MD5 hash of the file.
        Automatically assigned when save() is called.
The size of the file (in bytes).

Automatically assigned when save() is called.

Proxy class to facilitate access to all paper-related properties

Accepted revision

Call for proposals attr = u'cfp'

Can comment (user, check state=False)

Can judge (user, check state=False)

Can manage (user)

Can review (user, check state=False)

Can submit (user)

Create comment endpoint = u'papers.submit_comment'

Create judgment endpoint = u'papers.judge_paper'

Create review endpoint = u'papers.submit_review'

Delete comment endpoint = u'papers.delete_comment'

Edit comment endpoint = u'papers.edit_comment'

Edit review endpoint = u'papers.edit_review'

Event

Files

Get last revision()

Get revisions()

Is in final state

Judgment comment

Last revision

Locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:
The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
class indico.modules.events.papers.models.review_questions.PaperReviewQuestion(**kwargs):
    Bases: indico.core.db.sqlalchemy.review_questions.ReviewQuestionMixin,
           sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from `kwargs`.
    Sets attributes on the constructed instance using the names and values in `kwargs`.
    Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

    description
    event
    event_backref_name = u'paper_review_questions'
    event_id
    field
    field_data
    field_type
    id
    is_deleted
    is_required
    locator

    Defines a smart locator property.
    This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
```
This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}
@locator.other
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```position
```
```title
```
```type
```
```class
indico.modules.events.papers.models.review_ratings.PaperReviewRating(**kwargs)
Bases: indico.core.db.sqlalchemy.review_ratings.ReviewRatingMixin, sqlalchemy.ext.declarative.api.Model
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
```id
```question
```question_class
alias of indico.modules.events.papers.models.review_questions.PaperReviewQuestion
```question_id
```review
```review_class
alias of indico.modules.events.papers.models.reviews.PaperReview
```review_id
```value
```class
indico.modules.events.papers.models.reviews.PaperAction
Bases: indico.util.struct.enum.RichIntEnum
accept = 1
reject = 2
to_be_corrected = 3
```class
indico.modules.events.papers.models.reviews.PaperCommentVisibility
Bases: indico.util.struct.enum.RichIntEnum
Most to least restrictive visibility for paper comments
```
class indico.modules.events.papers.models.reviews.PaperJudgmentProxy(paper)
    Bases: object
    Represents a timeline item for the non final judgments

class indico.modules.events.papers.models.reviews.PaperReview(**kwargs)
    Bases: indico.modules.events.models.reviews.ProposalReviewMixin, indico.core.db.sqlalchemy.descriptions.RenderModeMixin, sqlalchemy.ext.declarative.api.Model
    Represents a paper review, emitted by a layout or content reviewer
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    TIMELINE_TYPE = u'review'
    can_edit (user, check_state=False)
    can_view(user)
    comment
    created_dt
    default_render_mode = 2
    group_attr = u'type'
    group_proxy_cls
        alias of PaperTypeProxy

id

locator
    Defines a smart locator property.
    This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
    This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.
    If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}
```

(locates on next page)
def locator(self):
    return {...}

The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
modified_dt
possible_render_modes = set([<RenderMode.markdown: 2>])
proposed_action
render_mode = 2
revision
revision_attr = u'revision'
revision_id
score
type
user
user_id
visibility
class indico.modules.events.papers.models.reviews.PaperReviewType
    Bases: indico.util.struct.enum.RichIntEnum
    content = 2
    layout = 1

class indico.modules.events.papers.models.reviews.PaperTypeProxy(group)
    Bases: indico.modules.events.models.reviews.ProposalGroupProxy

class indico.modules.events.papers.models.reviews.PaperReviewType
    Bases: indico.util.struct.enum.RichIntEnum
    content = 2
    layout = 1

    def locator(self):
        return {...}

    @locator.other
    def locator(self):
        return {...}
```

The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.
class indico.modules.events.papers.models.revisions.PaperRevision(*args, **kwargs):
    Bases: indico.modules.events.models.reviews.ProposalRevisionMixin,
           indico.core.db.sqlalchemy.descriptions.RenderModeMixin,
           sqlalchemy.ext.declarative.api.Model
    default_render_mode = 2
    get_reviewed_for_groups(user, include_reviewed=False)
    get_reviews(group=None, user=None)
    get_spotlight_file()
    get_timeline(user=None)
    has_userReviewed(user, review_type=None)
    id
    isLastRevision
    judge
    judge_id
    judgmentComment
    judgmentDt
    locator
        Defines a smart locator property.
        This behaves pretty much like a normal read-only property and the decorated function should return a dict
        containing the necessary data to build a URL for the object.
        This decorator should usually be applied to a method named locator as this name is required for
        get_locator to find it automatically when just passing the object.
        If you need more than one locator, you can define it like this:

        @locator_property
        def locator(self):
            return {...}
        @locator.other
        def locator(self):
            return {...}

    The other locator can then be accessed by passing obj.locator.other to the code expecting an
    object with a locator.
    number
    paper
    possibleRenderModes = set([<RenderMode.markdown: 2>])
    proposalAttr = u'paper'
    renderMode = 2
    spotlightFile
    state
    submittedDt
submitter
submitter_id
timeline
class indico.modules.events.papers.models.revisions.PaperRevisionState
    Bases: indico.util.struct.enum.RichIntEnum
    accepted = 2
    rejected = 3
    submitted = 1
    to_be_corrected = 4
class indico.modules.events.papers.models.templates.PaperTemplate(**kwargs)
    Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

add_file_date_column = False
content_type
    The MIME type of the file
created_dt = None
description
event
event_id
filename
    The name of the file
id
locator
    Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

5.1. API reference
The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

### md5
An MD5 hash of the file.
Automatically assigned when `save()` is called.

### name

### size
The size of the file (in bytes).
Automatically assigned when `save()` is called.

### storage_backend

### storage_file_id

## Operations

indico.modules.events.papers.operations.close_cfp(event)

indico.modules.events.papers.operations.create_comment(*args, **kwargs)

indico.modules.events.papers.operations.create_competences(event, user, competences)

indico.modules.events.papers.operations.create_paper_revision(paper, submitter, files)

indico.modules.events.papers.operations.create_paper_template(event, data)

indico.modules.events.papers.operations.create_review(paper, review_type, user, review_data, questions_data)

indico.modules.events.papers.operations.delete_comment(comment)

indico.modules.events.papers.operations.delete_paper_template(template)

indico.modules.events.papers.operations.judge_paper(*args, **kwargs)

indico.modules.events.papers.operations.open_cfp(event)

indico.modules.events.papers.operations.reset_paper_state(paper)

indico.modules.events.papers.operations.schedule_cfp(event, start_dt, end_dt)

indico.modules.events.papers.operations.set_deadline(event, role, deadline, enforce=True)

indico.modules.events.papers.operations.set_reviewing_state(event, reviewing_type, enable)

indico.modules.events.papers.operations.update_comment(comment, text, visibility)

indico.modules.events.papers.operations.update_competences(user_competences, competences)

indico.modules.events.papers.operations.update_paper_template(template, data)

indico.modules.events.papers.operations.update_review(review, review_data, questions_data)

indico.modules.events.papers.operations.update_reviewing_roles(event, users, contributions, role, assign)
indico.modules.events.papers.operations.update_team_members(event, managers, judges, content_reviewers=None, layout_reviewers=None)

Utilities

indico.modules.events.papers.util.get_contributions_with_paper_submitted_by_user(event, user)

indico.modules.events.papers.util.get_events_with_paper_roles(user, dt=None)
Get the IDs and PR roles of events where the user has any kind of paper reviewing privileges.

Parameters
- user – A User
- dt – Only include events taking place on/after that date

Returns A dict mapping event IDs to a set of roles

indico.modules.events.papers.util.get_user_contributions_to_review(event, user)
Get the list of contributions where user has paper to review

indico.modules.events.papers.util.get_user_reviewed_contributions(event, user)
Get the list of contributions where user already reviewed paper

indico.modules.events.papers.util.get_user_submittable_contributions(event, user)

indico.modules.events.papers.util.has_contributions_with_user_paper_submission_rights(event, user)

Settings

class indico.modules.events.settings.EventACLProxy(proxy)
Bases: indico.core.settings.proxy.ACLProxyBase
Proxy class for event-specific ACL settings

add_principal(event, *args, **kwargs)
Adds a principal to an ACL

Parameters
- event – Event (or its ID)
- name – Setting name
- principal – A User or a GroupProxy

contains_user(event, *args, **kwargs)
Checks if a user is in an ACL.
To pass this check, the user can either be in the ACL itself or in a group in the ACL.

Parameters
- event – Event (or its ID)
- name – Setting name
• **user** – A *User*

**get** *(event, *args, **kwargs)*
Retrieves an ACL setting

**Parameters**
- **event** – Event (or its ID)
- **name** – Setting name

**merge_users** *(target, source)*
Replaces all ACL user entries for source with target

**remove_principal** *(event, *args, **kwargs)*
Removes a principal from an ACL

**Parameters**
- **event** – Event (or its ID)
- **name** – Setting name
- **principal** – A *User* or a *GroupProxy*

**set** *(event, *args, **kwargs)*
Replaces an ACL with a new one

**Parameters**
- **event** – Event (or its ID)
- **name** – Setting name
- **acl** – A set containing principals (users/groups)

```python
class indico.modules.events.settings.EventSettingProperty(proxy, name, default=<object object>, attr=None)
```

**Bases:** indico.core.settings.proxy.SettingProperty

**attr** = u'event'

```python
class indico.modules.events.settings.EventSettingsProxy(module, defaults=None, strict=True, acls=None, converters=None)
```

**Bases:** indico.core.settings.proxy.SettingsProxyBase

Proxy class to access event-specific settings for a certain module

**acl_proxy_class**
alias of *EventACLProxy*

**delete** *(event, *args, **kwargs)*
Deletes settings.

**Parameters**
- **event** – Event (or its ID)
- **names** – One or more names of settings to delete

**delete_all** *(event, *args, **kwargs)*
Deletes all settings.

**Parameters**
- **event** – Event (or its ID)
get(event, *args, **kwargs)
Retrieves the value of a single setting.

Parameters
- event – Event (or its ID)
- name – Setting name
- default – Default value in case the setting does not exist

Returns The settings’s value or the default value

get_all(event, *args, **kwargs)
Retrieves all settings

Parameters
- event – Event (or its ID)
- no_defaults – Only return existing settings and ignore defaults.

Returns Dict containing the settings

query
Returns a query object filtering by the proxy’s module.

set(event, *args, **kwargs)
Sets a single setting.

Parameters
- event – Event (or its ID)
- name – Setting name
- value – Setting value; must be JSON-serializable

set_multi(event, *args, **kwargs)
Sets multiple settings at once.

Parameters
- event – Event (or its ID)
- items – Dict containing the new settings

class indico.modules.events.settings.ThemeSettingsProxy
Bases: object

defaults
get_themes_for(**kwargs)

settings

themes

indico.modules.events.settings.event_or_id(f)

5.1.11 Payment

Todo: Docstrings (module, models, plugins)
Models

**exception** indico.modules.events.payment.models.transactions.DoublePaymentTransaction
   Bases: exceptions.Exception

**exception** indico.modules.events.payment.models.transactions.IgnoredTransactionAction
   Bases: exceptions.Exception

**exception** indico.modules.events.payment.models.transactions.InvalidManualTransactionAction
   Bases: exceptions.Exception

**exception** indico.modules.events.payment.models.transactions.InvalidTransactionAction
   Bases: exceptions.Exception

**exception** indico.modules.events.payment.models.transactions.InvalidTransactionStatus
   Bases: exceptions.Exception

**class** indico.modules.events.payment.models.transactions.PaymentTransaction(**kwargs)
   Bases: sqlalchemy.ext.declarative.api.Model

Payment transactions
   A simple constructor that allows initialization from kwargs.
   Sets attributes on the constructed instance using the names and values in kwargs.
   Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

- **amount**
  the base amount the user needs to pay (without payment-specific fees)

- **currency**
  the currency of the payment (ISO string, e.g. EUR or USD)

- **data**
  plugin-specific data of the payment

- **id**
  Entry ID

- **is_manual**

- **plugin**

- **provider**
  the provider of the payment (e.g. manual, PayPal etc.)

- **registration**
  The associated registration

- **registration_id**
  ID of the associated registration

- **render_details()**
  Renders the transaction details

- **status**
  a TransactionStatus

- **timestamp**
  the date and time the transaction was recorded
class indico.modules.events.payment.models.transactions.TransactionAction
    Bases: int, indico.util.struct.enum.IndicoEnum
    cancel = 2
    complete = 1
    pending = 3
    reject = 4

class indico.modules.events.payment.models.transactions.TransactionStatus
    Bases: int, indico.util.struct.enum.IndicoEnum
    cancelled = 2
        payment cancelled manually
    failed = 3
        payment attempt failed
    pending = 4
        payment on hold pending approval of merchant
    rejected = 5
        payment rejected after being pending
    successful = 1
        payment attempt succeeded

class indico.modules.events.payment.models.transactions.TransactionStatusTransition
    Bases: object
    initial_statuses = [<TransactionStatus.cancelled: 2>, <TransactionStatus.failed: 3>,
    classmethod next (transaction, action, provider=None)

Utilities

indico.modules.events.payment.util.get_active_payment_plugins (event)
    Returns a dict containing the active payment plugins of an event.
indico.modules.events.payment.util.get_payment_plugins ()
    Returns a dict containing the available payment plugins.
indico.modules.events.payment.util.register_transaction (registration, amount, currency, action,
    provider=None, data=None)

    Creates a new transaction for a certain transaction action.

    Parameters
    • registration – the Registration associated to the transaction
    • amount – the (strictly positive) amount of the transaction
    • currency – the currency used for the transaction
    • action – the TransactionAction of the transaction
    • provider – the payment method name of the transaction, or '_manual' if no payment
        method has been used
    • data – arbitrary JSON-serializable data specific to the transaction’s provider
Plugins

class indico.modules.events.payment.plugins.PaymentPluginMixin
Bases: object

adjust_payment_form_data(data)
Updates the payment form data if necessary.

This method can be overridden to update e.g. the amount based on choices the user makes in the payment
form or to provide additional data to the form. To do so, data must be modified.

Parameters
data – a dict containing event, registration, amount, currency, settings and
event_settings

can_be_modified(user, event)
Checks if the user is allowed to enable/disable/modify the payment method.

Parameters

• user – the User representing the user
• event – the Event

category = u'Payment'
default_settings
event_settings_form
    alias of PaymentEventSettingsFormBase

get_event_management_url(event, **kwargs)
get_invalid_regforms(event)
    Return registration forms with incompatible currencies

get_method_name(event)
    Returns the (customized) name of the payment method.

init()

logo_url

render_payment_form(registration)
    Returns the payment form shown to the user.

    Parameters
    registration – a Registration object

render_transaction_details(transaction)
    Renders the transaction details in event management

    Override this (or inherit from the template) to show more useful data such as transaction IDs

    Parameters
    transaction – the PaymentTransaction

settings_form
    alias of PaymentPluginSettingsFormBase

supports_currency(currency)

valid_currencies = None
    Set containing all valid currencies. Set to None to allow all.
5.1.12 Person

Todo: Docstrings (module, operations)

Operations

```
indico.modules.events.persons.operations.update_person(person, data)
```

Placeholders

```
class indico.modules.events.persons.placeholders.EmailPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'Email of the person'
    name = u'email'
    classmethod render(person, event, **kwargs)

class indico.modules.events.persons.placeholders.EventLinkPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'Link to the event'
    name = u'event_link'
    classmethod render(person, event, **kwargs)

class indico.modules.events.persons.placeholders.EventTitlePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The title of the event'
    name = u'event_title'
    classmethod render(person, event, **kwargs)

class indico.modules.events.persons.placeholders.FirstNamePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'First name of the person'
    name = u'first_name'
    classmethod render(person, event, **kwargs)

class indico.modules.events.persons.placeholders.LastNamePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'Last name of the person'
    name = u'last_name'
    classmethod render(person, event, **kwargs)

class indico.modules.events.persons.placeholders.RegisterLinkPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The link for the registration page'
    name = u'register_link'
```
5.1.13 Registration

Todo: Docstrings (module, models, utilities, statistics)

Models

class indico.modules.events.registration.models.registrations.Registration(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    Somebody’s registration for an event through a registration form
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    base_price
    The base registration fee (that is not specific to form items)

    billable_data

    can_be_modified

    checked_in
    Whether the person has checked in. Setting this also sets or clears checked_in_dt.

    checked_in_dt
    The date/time when the person has checked in

    currency
    Registration price currency

    data
    The registration this data is associated with

    data_by_field

    display_full_name
    Return the full name using the user’s preferred name format.

    email
    The email of the registrant

    event
    The Event containing this registration

    event_id
    The ID of the event

    first_name
    The first name of the registrant

    friendly_id
    The human-friendly ID for the object
full_name
Returns the user’s name in ‘Firstname Lastname’ notation.

classmethod get_all_for_event(event)
Retrieve all registrations in all registration forms of an event.

get_full_name(last_name_first=True, last_name_upper=False, abbrev_first_name=False)
Returns the user’s in the specified notation.
If not format options are specified, the name is returned in the ‘Lastname, Firstname’ notation.
Note: Do not use positional arguments when calling this method. Always use keyword arguments!

Parameters

• last_name_first – if “lastname, firstname” instead of “firstname lastname” should be used
• last_name_upper – if the last name should be all-uppercase
• abbrev_first_name – if the first name should be abbreviated to use only the first character

get_personal_data()

has_files

id
The ID of the object

is_active

is_cancelled

is_deleted
If the registration has been deleted

is_paid
Returns whether the registration has been paid for.

is_publishable

is_ticket_blocked
Check whether the ticket is blocked by a plugin

last_name
The last name of the registrant

locator
Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}
@locator.other
```

(continues on next page)
def locator(self):
    return {...}

The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```
order_by_name = (<sqlalchemy.sql.functions.Function at 0x7fcf6ecd5610; lower>,
<sqlalchemy.sql.functions.Function at 0x7fcf6ecd5650; lower>,
<sqlalchemy.orm.attributes.InstrumentedAttribute object>)
```

```
payment_dt
    The date/time when the registration has been paid for
```

```
price
    The total price of the registration.
    This includes the base price, the field-specific price, and the custom price adjustment for the registrant.
    
    Return type  Decimal
```

```
price_adjustment
    The price modifier applied to the final calculated price
```

```
registration_form_id
    The ID of the registration form
```

```
render_base_price()
render_price()
render_price_adjustment()
sections_with_answered_fields
state
    The state a registration is in
```

```
submitted_dt
    The date/time when the registration was recorded
```

```
summary_data
    Export registration data nested in sections and fields
```

```
sync_state(_skip_moderation=True)
    Sync the state of the registration
```

```
ticket_uuid
    The unique token used in tickets
```

```
transaction
    The latest payment transaction associated with this registration
```

```
transaction_id
    The ID of the latest payment transaction associated with this registration
```

```
update_state(approved=None, paid=None, rejected=None, _skip_moderation=False)
    Update the state of the registration for a given action
    
    The accepted kwargs are the possible actions. True means that the action occurred and False that it was reverted.
```

```
user
```

```
user_id
    The ID of the user who registered
```
uuid
The unguessable ID for the object

class indico.modules.events.registration.models.registrations.RegistrationData(**kwargs)
Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model

Data entry within a registration for a field in a registration form
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

add_file_date_column = False

content_type
The MIME type of the file

created_dt = None

data
The submitted data for the field

field_data
The associated field data object

field_data_id
The ID of the field data

file

file_required = False

filename
The name of the file

friendly_data

get_friendly_data(**kwargs)

locator
Defines a smart locator property.
This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.
If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

5.1. API reference
md5
An MD5 hash of the file.
Automatically assigned when \texttt{save()} is called.

\texttt{price}

\texttt{registration_id}
The ID of the registration

\texttt{render_price()}

\texttt{search_data}

\texttt{size}
The size of the file (in bytes).
Automatically assigned when \texttt{save()} is called.

\texttt{storage_backend}

\texttt{storage_file_id}

\texttt{summary_data}

\texttt{user_data}

\texttt{class \texttt{indico.modules.events.registration.models.registrations.RegistrationState}}
Bases: \texttt{indico.util.struct.enum.RichIntEnum}

complete = 1
pending = 2
rejected = 3
unpaid = 5
withdrawn = 4

\texttt{class \texttt{indico.modules.events.registration.models.form_fields.RegistrationFormField}(**\texttt{kwargs})}
Bases: \texttt{indico.modules.events.registration.models.items.RegistrationFormItem}

A registration form field

A simple constructor that allows initialization from \texttt{kwargs}.

Sets attributes on the constructed instance using the names and values in \texttt{kwargs}.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

\texttt{calculate_price} (registration_data)

\texttt{children}

\texttt{current_data}

\texttt{current_data_id}

\texttt{data}

\texttt{data_versions}

\texttt{description}

\texttt{field_impl}

Gets the implementation of the field.
Returns An instance of a RegistrationFormFieldBase subclass

get_friendly_data(registration_data, **kwargs)

html_field_name
id
input_type
is_deleted
is_enabled
is_manager_only
is_required
locator
parent_id
personal_data_type
position
registration_form_id
title
type
versioned_data
view_data

class indico.modules.events.registration.models.form_fields.RegistrationFormFieldData(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Model

Description of a registration form field

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

field_id
    The ID of the registration form field

id
    The ID of the object

versioned_data
    Data describing the field

class indico.modules.events.registration.models.form_fields.RegistrationFormPersonalDataField
Bases: indico.modules.events.registration.models.form_fields.RegistrationFormField

RegistrationFormField

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

children
current_data
current_data_id
data
data_versions
description
html_field_name
id
input_type
is_deleted
is_enabled
is_manager_only
is_required
parent_id
personal_data_type
position
registration_form_id
title
type
view_data
class indico.modules.events.registration.models.forms.ModificationMode
    Bases: indico.util.struct.enum.RichIntEnum

    allowed_always = 1
    allowed_until_payment = 2
    not_allowed = 3

class indico.modules.events.registration.models.forms.RegistrationForm(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A registration form for an event
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

active_fields
active_registrations
base_price
    The base fee users have to pay when registering

can_submit(user)
contact_info
    Contact information for registrants
currency
    Currency for prices in the registration form

disabled_sections

date
    Datetime when the registration form is closed

event
    The Event containing this registration form

event_id
    The ID of the event

form_items

get_personal_data_field_id(personal_data_type)
    Returns the field id corresponding to the personal data field with the given name.

get_registration(**kwargs)
    Retrieves registrations for this registration form by user or uuid

has_ended

has_started

id
    The ID of the object

introduction

invitations
    The registration invitations associated with this form

is_active

is_deleted
    Whether the registration has been marked as deleted

is_modification_allowed(registration)
    Checks whether a registration may be modified

is_modification_open

is_open

is_participation
    Whether it’s the ‘Participants’ form of a meeting/lecture

is_scheduled

limit_reached

locator

manager_notification_recipients
    List of emails that should receive management notifications

manager_notifications_enabled
    Whether the manager notifications for this event are enabled

message_complete
    Custom message to include in emails for complete registrations

message_pending
    Custom message to include in emails for pending registrations

5.1. API reference
message_unpaid
   Custom message to include in emails for unpaid registrations

moderation_enabled
   Whether registrations must be approved by a manager

modification_end_dt
   Datetime when the modification period is over

modification_mode
   Whether registration modifications are allowed

notification_sender_address
   Notifications sender address

publish_checkin_enabled
   Whether checked-in status should be displayed in the event pages and participant list

publish_registration_count
   Whether to display the number of registrations

publishRegistrations_enabled
   Whether registrations should be displayed in the participant list

registration_limit
   Maximum number of registrations allowed

registrations
   The registrations associated with this form

render_base_price()

require_login
   Whether users must be logged in to register

require_user
   Whether registrations must be associated with an Indico account

sections

sender_address

start_dt
   Datetime when the registration form is open

ticket_on_email
   Whether to send tickets by e-mail

ticket_on_event_page
   Whether to show a ticket download link on the event homepage

ticket_on_summary_page
   Whether to show a ticket download link on the registration summary page

ticket_template
   The template used to generate tickets

ticket_template_id
   The ID of the template used to generate tickets

tickets_enabled
   Whether tickets are enabled for this form

title
   The title of the registration form
class indico.modules.events.registration.models.invitations.InvitationState
    Bases: indico.util.struct.enum.RichIntEnum
    accepted = 1
    declined = 2
    pending = 0

class indico.modules.events.registration.models.invitations.RegistrationInvitation(**kwars)
    Bases: sqlalchemy.ext.declarative.api.Model
    An invitation for someone to register
    A simple constructor that allows initialization from kwars.
    Sets attributes on the constructed instance using the names and values in kwars.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.

    affiliation
        The affiliation of the invited person

    email
        The email of the invited person

    first_name
        The first name of the invited person

    id
        The ID of the invitation

    last_name
        The last name of the invited person

    locator
        Defines a smart locator property.
        This behaves pretty much like a normal read-only property and the decorated function should return a dict
        containing the necessary data to build a URL for the object.
        This decorator should usually be applied to a method named locator as this name is required for
        get_locator to find it automatically when just passing the object.
        If you need more than one locator, you can define it like this:

        ```python
        @locator_property
        def locator(self):
            return {...}

        @locator.other
        def locator(self):
            return {...}
        ```
        The other locator can then be accessed by passing obj.locator.other to the code expecting an
        object with a locator.

    registration
        The associated registration

    registration_form_id
        The ID of the registration form

5.1. API reference
registration_id
  The ID of the registration (if accepted)

skip_moderation
  Whether registration moderation should be skipped

state
  The state of the invitation

uuid
  The UUID of the invitation

class indico.modules.events.registration.models.items.PersonalDataType
  Bases: int, indico.util.struct.enum.IndicoEnum

  Description of the personal data items that exist on every registration form

  FIELD_DATA = [(<PersonalDataType.title: 5>, {u'input_type': u'single_choice', u'data': {u'item_type': u'dropdown', ... 9>, {u'is_enabled': False, u'input_type': u'text', u'position': 1003, u'title': u'Position'})

  address = 6
  affiliation = 4

  column
  The Registration column in which the value is stored in addition to the regular registration data entry.

country = 8
email = 1
first_name = 2
get_title()
is_required
last_name = 3
phone = 7
position = 9
title = 5

class indico.modules.events.registration.models.items.RegistrationFormItem(**kwargs)
  Bases: sqlalchemy.ext.declarative.api.Model

  Generic registration form item

  A simple constructor that allows initialization from kwargs.

  Sets attributes on the constructed instance using the names and values in kwargs.

  Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

  children

  current_data
  The latest value of the field

  current_data_id
  The ID of the latest data

  data
  unversioned field data
data_versions
   The list of all versions of the field data

description
   Description of this field

id
   The ID of the object

input_type
   input type of this field

is_deleted
   Whether field has been “deleted”

is_enabled
   Whether the field is enabled

is_field

is_manager_only
   if the section is only accessible to managers

is_required
   determines if the field is mandatory

is_section

is_visible

parent_id
   The ID of the parent form item

personal_data_type
   The type of a personal data field

position

registration_form_id
   The ID of the registration form

title
   The title of this field

type
   The type of the registration form item

view_data
   Returns object with data that Angular can understand

class indico.modules.events.registration.models.items.RegistrationFormItemType
   Bases: int, indico.util.struct.enum.IndicoEnum

   field = 2
   field_pd = 5
   section = 1
   section_pd = 4
   text = 3

class indico.modules.events.registration.models.items.RegistrationFormPersonalDataSection
   Bases: indico.modules.events.registration.models.items.RegistrationFormSection
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
class RegistrationFormSection(**kwargs):
    Bases: indico.modules.events.registration.models.items.RegistrationFormItem
```

Registration form section that can contain fields and text

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
fields
id
input_type
is_deleted
is_enabled
is_manager_only
is_required
locator
own_data
parent_id
personal_data_type
position
registration_form_id
title
type
view_data

class indico.modules.events.registration.models.items.RegistrationFormText(**kwargs)
Bases: indico.modules.events.registration.models.items.RegistrationFormItem

Text to be displayed in registration form sections
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

children
current_data
current_data_id
data
data_versions
description
id
input_type
is_deleted
is_enabled
is_manager_only
is_required
locator
parent_id
Utilities

indico.modules.events.registration.util.build_registration_api_data(registration)

Checks whether an email address is suitable for registration.

Parameters

- **regform** – The registration form
- **email** – The email address
- **registration** – The existing registration (in case of modification)
- **management** – If it’s a manager adding a new registration

indico.modules.events.registration.util.create_personal_data_fields(regform)

Creates the special section/fields for personal data.

indico.modules.events.registration.util.create_registration(*args, **kwargs)

indico.modules.events.registration.util.generate_spreadsheet_from_registrations(registrations, regform_items, static_items)

Generates a spreadsheet data from a given registration list.

Parameters

- **registrations** – The list of registrations to include in the file
- **regform_items** – The registration form items to be used as columns
- **static_items** – Registration form information as extra columns

indico.modules.events.registration.util.generate_ticket(registration)

Generate a Pillow Image with a QR Code encoding a check-in ticket.

Parameters **registration** – corresponding Registration object

indico.modules.events.registration.util.get_event_regforms(event, user, with_registrations=False)

Get registration forms with information about user registrations.
Parameters

- **event** – the Event to get registration forms for
- **user** – A User
- **with_registrations** – Whether to return the user’s registration instead of just whether they have one

```python
indico.modules.events.registration.util.get_event_regforms_registrations(event, user, include_scheduled=True)
```

Get regforms and the associated registrations for an event+user.

Parameters

- **event** – the Event to get registration forms for
- **user** – A User
- **include_scheduled** – Whether to include scheduled but not open registration forms

Returns

A tuple, which includes:
- All registration forms which are scheduled, open or registered.
- A dict mapping all registration forms to the user’s registration if they have one.

```python
indico.modules.events.registration.util.get_event_section_data(regform, management=False, registration=None)
```

```python
indico.modules.events.registration.util.get_events_registered(user, dt=None)
```

Gets the IDs of events where the user is registered.

Parameters

- **user** – A User
- **dt** – Only include events taking place on/after that date

Returns
A set of event ids

```python
indico.modules.events.registration.util.get_published_registrations(event)
```

Get a list of published registrations for an event.

Parameters **event** – the Event to get registrations for

Returns
list of Registration objects

```python
indico.modules.events.registration.util.get_registered_event_persons(event)
```

Get all registered EventPersons of an event.

```python
indico.modules.events.registration.util.get_registrations_with_tickets(user, event)
```

```python
indico.modules.events.registration.util.get_ticket_attachments(registration)
```

```python
indico.modules.events.registration.util.get_title_uuid(regform, title)
```

Convert a string title to its UUID value

If the title does not exist in the title PD field, it will be ignored and returned as None.
Import event registrants from a CSV file into a form.

```
indico.modules.events.registration.util.import_registrations_from_csv(regform, fileobj, skip_moderation=True, notify_users=False)
```

Creates a WTForm based on registration form fields

```
indico.modules.events.registration.util.make_registration_form(regform, management=False, registration=None)
```

Update positions when deleting/disabling an item in order to prevent gaps

```
indico.modules.events.registration.util.update_regform_item_positions(regform)
```

Converts a flask-style rule to angular style

```
indico.modules.events.registration.util.url_rule_to_angular(endpoint)
```

### Placeholders

**class** `indico.modules.events.registration.placeholders.registrations.EventLinkPlaceholder`

<table>
<thead>
<tr>
<th>Bases:</th>
<th>indico.util.placeholders.Placeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>lu'Link to the event'</td>
</tr>
<tr>
<td>name</td>
<td>u'event_link'</td>
</tr>
<tr>
<td>classmethod</td>
<td><code>render(regform, registration)</code></td>
</tr>
</tbody>
</table>

**class** `indico.modules.events.registration.placeholders.registrations.EventTitlePlaceholder`

<table>
<thead>
<tr>
<th>Bases:</th>
<th>indico.util.placeholders.Placeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>lu'The title of the event'</td>
</tr>
<tr>
<td>name</td>
<td>u'event_title'</td>
</tr>
<tr>
<td>classmethod</td>
<td><code>render(regform, registration)</code></td>
</tr>
</tbody>
</table>

**class** `indico.modules.events.registration.placeholders.registrations.FieldPlaceholder`

<table>
<thead>
<tr>
<th>Bases:</th>
<th>indico.util.placeholders.ParametrizedPlaceholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>advanced</td>
<td>True</td>
</tr>
<tr>
<td>description</td>
<td>None</td>
</tr>
<tr>
<td>classmethod</td>
<td><code>iter_param_info(regform, registration)</code></td>
</tr>
<tr>
<td>name</td>
<td>u'field'</td>
</tr>
<tr>
<td>param_required</td>
<td>True</td>
</tr>
<tr>
<td>param_restricted</td>
<td>True</td>
</tr>
<tr>
<td>classmethod</td>
<td><code>render(param, regform, registration)</code></td>
</tr>
</tbody>
</table>

**class** `indico.modules.events.registration.placeholders.registrations.FirstNamePlaceholder`

<table>
<thead>
<tr>
<th>Bases:</th>
<th>indico.util.placeholders.Placeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>lu'First name of the person'</td>
</tr>
<tr>
<td>name</td>
<td>u'first_name'</td>
</tr>
</tbody>
</table>
classmethod render(regform, registration)
class indico.modules.events.registration.placeholders.registrations.IDPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The ID of the registration'
    name = u'id'
    classmethod render(regform, registration)
class indico.modules.events.registration.placeholders.registrations.LastNamePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'Last name of the person'
    name = u'last_name'
    classmethod render(regform, registration)
class indico.modules.events.registration.placeholders.registrations.LinkPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'The link to the registration details'
    name = u'link'
    classmethod render(regform, registration)
class indico.modules.events.registration.placeholders.invitations.FirstNamePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'First name of the person'
    name = u'first_name'
    classmethod render(invitation)
class indico.modules.events.registration.placeholders.invitations.InvitationLinkPlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'Link to accept/decline the invitation'
    name = u'invitation_link'
    classmethod render(invitation)
    required = True
class indico.modules.events.registration.placeholders.invitations.LastNamePlaceholder
    Bases: indico.util.placeholders.Placeholder
    description = lu'Last name of the person'
    name = u'last_name'
    classmethod render(invitation)
Settings

class indico.modules.events.registration.settings.RegistrationSettingsProxy (module, defaults=None, strict=True, acls=None, converters=None)

Bases: indico.modules.events.settings.EventSettingsProxy

Store per-event registration settings

get_participant_list_columns (event, form=None)

get_participant_list_form_ids (event)

set_participant_list_columns (event, columns, form=None)

set_participant_list_form_ids (event, form_ids)

Statistics

class indico.modules.events.registration.stats.AccommodationStats (field)

Bases: indico.modules.events.registration.stats.FieldStats, indico.modules.events.registration.stats.StatsBase

class indico.modules.events.registration.stats.Cell

Bases: indico.modules.events.registration.stats.Cell

Hold data and type for a cell of a stats table

The table below indicates the valid types and expected data.

<table>
<thead>
<tr>
<th>type</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>str – string value</td>
</tr>
<tr>
<td>progress</td>
<td>(int, str) – a tuple with the progress (a value between 0 and 1) and a label</td>
</tr>
<tr>
<td>progress-stacked</td>
<td>([int], str) – a tuple with a list of progresses (values which must sum up to 1) and a label</td>
</tr>
<tr>
<td>currency</td>
<td>float – numeric value</td>
</tr>
<tr>
<td>icon</td>
<td>str – icon name from _icons.scss</td>
</tr>
<tr>
<td>default</td>
<td>None – renders a default cell with an — (use Cell(type='str') for an empty cell)</td>
</tr>
</tbody>
</table>

Parameters

• type – str – The type of data in the cell
• data – The data for the cell
• colspan – int – HTML colspan value for the cell
• classes – [str] – HTML classes to apply to the cell
• qtip – str – content for qtip

class indico.modules.events.registration.stats.DataItem

Bases: indico.modules.events.registration.stats.DataItem

Holds the aggregation of some data, intended for stats tables as a aggregation from which to generate cells.
Parameters

• **regs** – int – number of registrant
• **attendance** – int – number of people attending
• **capacity** – int – maximum number of people allowed to attend (0 if unlimited)
• **billable** – bool – whether the item is billable to the or not
• **cancelled** – bool – whether the item is cancelled or not
• **price** – str – the price of the item
• **fixed_price** – bool – True if the price is per registrant, False if accompanying guests must pay as well.
• **paid** – int – number of registrants who paid
• **paid_amount** – float – amount already paid by registrants
• **unpaid** – int – number of registrants who haven’t paid
• **unpaid_amount** – float – amount not already paid by registrants

class indico.modules.events.registration.stats.FieldStats(field, **kwargs)
Bases: object
Holds stats for a registration form field

def get_table()
    Returns a table containing the stats for each item.

    Returns dict – A table with a list of head cells (key: ‘head’) and a list of rows (key: ‘rows’) where each row is a list of cells.

    is_currency_shown

class indico.modules.events.registration.stats.OverviewStats(regform)
Bases: indico.modules.events.registration.stats.StatsBase
Generic stats for a registration form

class indico.modules.events.registration.stats.StatsBase(title, subtitle, type, **kwargs)
Bases: object
Base class for registration form statistics

Parameters

• **title** – str – the title for the stats box
• **subtitle** – str – the subtitle for the stats box
• **type** – str – the type used in Jinja to display the stats

is_currency_shown

5.1.14 Reminder

Todo: Docstrings (module)
Models

class indico.modules.events.reminders.models.reminders.EventReminder(**kwargs):
    Bases: sqlalchemy.ext.declarative.api.Model

    Email reminders for events
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.

    **kwargs
        Returns all recipients of the notifications.
        This includes both explicit recipients and, if enabled, participants of the event.

    created_dt
        The date/time when the reminder was created

    creator
        The user who created the reminder

    creator_id
        The ID of the user who created the reminder

    event
        The Event this reminder is associated with

    event_id
        The ID of the event

    event_start_delta
        How long before the event start the reminder should be sent This is needed to update the scheduled_dt
        when changing the start time of the event.

    id
        The ID of the reminder

    include_description
        If the notification should include the event’s description.

    include_summary
        If the notification should include a summary of the event’s schedule.

    is_overdue

    is_relative
        Returns if the reminder is relative to the event time

    is_sent
        If the reminder has been sent

    locator

    message
        Custom message to include in the email

    recipients
        The recipients of the notification

    reply_to_address
        The address to use as Reply-To in the notification email.
**scheduled_dt**

The date/time when the reminder should be sent

**send()**

Sends the reminder to its recipients.

**send_to_participants**

If the notification should also be sent to all event participants

### Utilities

```python
indico.modules.events.reminders.util.make_reminder_email(event, with_agenda, with_description, note)
```

Returns the template module for the reminder email.

**Parameters**

- **event** – The event
- **with_agenda** – If the event’s agenda should be included
- **note** – A custom message to include in the email

### 5.1.15 Request

**Todo:** Docstrings (module)

### Models

```python
class indico.modules.events.requests.models.requests.Request(**kwargs):
    Bases: sqlalchemy.ext.declarative.api.Model
```

Event-related requests, e.g. for a webcast

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

**can_be_modified**

Determines if the request can be modified or if a new one must be sent

**comment**

an optional comment for an accepted/rejected request

**created_by_id**

ID of the user creating the request

**created_by_user**

The user who created the request

**created_dt**

the date/time the request was created

**data**

plugin-specific data of the request
definition

**event**

The Event this agreement is associated with

**event_id**

ID of the event

**classmethod find_latest_for_event**(event, type_=None)

Returns the latest requests for a given event.

**Parameters**

- **event** – the event to find the requests for
- **type** – the request type to retrieve, or *None* to get all

**Returns** a dict mapping request types to a *Request* or if *type_* was specified, a single *Request* or *None*

**id**

request ID

**locator**

**processed_by_id**

ID of the user processing the request

**processed_by_user**

The user who processed the request

**processed_dt**

the date/time the request was accepted/rejected

**state**

the requests’s date, a *RequestState* value

**type**

the request type name

**class** indico.modules.events.requests.models.requests.*RequestState*

Bases: indico.util.struct.enum.RichIntEnum

- **accepted** = 1
- **pending** = 0
- **rejected** = 2
- **withdrawn** = 3

**Utilities**

- **indico.modules.events.requests.util.get_request_definitions()**
  Returns a dict of request definitions
- **indico.modules.events.requests.util.is_request_manager(user)**
  Checks if the user manages any request types
- **class** indico.modules.events.requests.base.*RequestDefinitionBase*

  Bases: *object*

A service request which can be sent by event managers.
classmethod **accept** *(req, data, user)*

Accept the request.

To ensure that additional data is saved, this method should call :method:`manager_save`.

**Parameters**

- **req** – the `Request` of the request
- **data** – the form data from the management form
- **user** – the user processing the request

classmethod **can_be_managed** *(user)*

Check whether the user is allowed to manage this request type.

**Parameters**

- **user** – a `User`

classmethod **create_form** *(event, existing_request=None)*

Create the request form.

**Parameters**

- **event** – the event the request is for
- **existing_request** – the `Request` if there’s an existing request of this type

**Returns** an instance of an `IndicoForm` subclass

classmethod **create_manager_form** *(req)*

Create the request management form.

**Parameters**

- **req** – the `Request` of the request

**Returns** an instance of an `IndicoForm` subclass

form = None

the `IndicoForm` to use for the request form

form_defaults = {}

default values to use if there’s no existing request

classmethod **get_manager_notification_emails** *

Return the email addresses of users who manage requests of this type.

The email addresses are used only for notifications. It usually makes sense to return the email addresses of the users who pass the :method:`can_be_managed` check.

**Returns** set of email addresses

classmethod **get_notification_reply_email** *

Return the `Reply-To` e-mail address for notifications.

classmethod **get_notification_template** *(name, **context)*

Get the template module for a notification email.

**Parameters**

- **name** – the template name
- **context** – data passed to the template

manager_form

the `IndicoForm` to use for the request manager form

alias of `RequestManagerForm`
classmethod manager_save (req, data)
Save management-specific data.

This method is called when the management form is submitted without accepting/rejecting the request (which is guaranteed to be already accepted or rejected).

Parameters
- req – the Request of the request
- data – the form data from the management form

name = None
the unique internal name of the request type

plugin = None
the plugin containing this request definition - assigned automatically

classmethod reject (req, data, user)
Reject the request.

To ensure that additional data is saved, this method should call :method:`manager_save`.

Parameters
- req – the Request of the request
- data – the form data from the management form
- user – the user processing the request

classmethod render_form (event, **kwargs)
Render the request form.

Parameters
- event – the event the request is for
- kwargs – arguments passed to the template

classmethod send (req, data)
Send a new/modified request.

Parameters
- req – the Request of the request
- data – the form data from the request form

title = None
the title of the request type as shown to users

classmethod withdraw (req, notify_event_managers=True)
Withdraw the request.

Parameters
- req – the Request of the request
- notify_event_managers – if event managers should be notified

5.1.16 Session

Todo: Docstrings (module, models, operations, utilities)
Models

class indico.modules.events.sessions.models.sessions.Session(**kwargs)
    Bases: indico.core.db.sqlalchemy.descriptions.DescriptionMixin, indico.core.
db.sqlalchemy.colors.ColorMixin, indico.core.db.sqlalchemy.protection.
ProtectionManagersMixin, indico.core.db.sqlalchemy.locations.LocationMixin,
indico.core.db.sqlalchemy.attachments.AttachedItemsMixin, indico.core.db.
db.sqlalchemy.notes.AttachedNotesMixin, sqlalchemy.ext.declarative.api.Model

ATTACHMENT_FOLDER_ID_COLUMN = u'session_id'
PRELOAD_EVENT_ATTACHED_ITEMS = True
PRELOAD_EVENT_NOTES = True
access_key = None
acl_entries
allow_relationship_preloading = True
background_color
blocks

  can_manage_blocks (user, allow_admin=True)
      Check whether a user can manage session blocks.
      
      This only applies to the blocks themselves, not to contributions inside them.

  can_manage_contributions (user, allow_admin=True)
      Check whether a user can manage contributions within the session.

code
conveners

default_colors = ColorTuple(text=u'202020', background=u'e3f2d3')

default_contribution_duration

default_render_mode = 2

disallowed_render_modes = frozenset([])
end_dt

event

event_id

friendly_id
      The human-friendly ID for the session

get_non_inheriting_objects ()
      Get a set of child objects that do not inherit protection

id

inherit_location

inheriting_have_acl = True

is_deleted

is_poster
location_backref_name = u'sessions'

location_parent

locator
   Defines a smart locator property.

   This behaves pretty much like a normal read-only property and the decorated function should return a dict
   containing the necessary data to build a URL for the object.

   This decorator should usually be applied to a method named locator as this name is required for
   get_locator to find it automatically when just passing the object.

   If you need more than one locator, you can define it like this:

   ```python
   @locator_property
   def locator(self):
       return {...}
   @locator.other
   def locator(self):
       return {...}
   ```

   The other locator can then be accessed by passing obj.locator.other to the code expecting an
   object with a locator.

own_address

own_no_access_contact = None

own_room

own_room_id

own_room_name

own_venue

own_venue_id

own_venue_name

possible_render_modes = set([<RenderMode.markdown: 2>])

classmethod preload_acl_entries(event)

protection_mode

protection_parent

render_mode = 2

session
   Convenience property so all event entities have it

start_dt

text_color

title

type

type_id
class indico.modules.events.sessions.models.blocks.SessionBlock(**kwargs)
    Bases: indico.core.db.sqlalchemy.locations.LocationMixin, sqlalchemy.ext.declarative.api.Model

    can_access (user, allow_admin=True)
    can_edit_note (user)
    can_manage (user, allow_admin=True)
    can_manage_attachments (user)
    code
    contribution_count
    duration
    end_dt
    event
    full_title
    has_note
    id
    inherit_location
    location_backref_name = u'session_blocks'
    location_parent

    locator
    Defines a smart locator property.
    This behaves pretty much like a normal read-only property and the decorated function should return a dict
    containing the necessary data to build a URL for the object.
    This decorator should usually be applied to a method named locator as this name is required for
    get_locator to find it automatically when just passing the object.
    If you need more than one locator, you can define it like this:

    @locator_property
    def locator(self):
        return {...}

    @locator.other
    def locator(self):
        return {...}

    The other locator can then be accessed by passing objlocator.other to the code expecting an
    object with a locator.

    note
    own_address
    own_room
    own_room_id
    own_room_name
    own_venue
own_venue_id
own_venue_name
person_links
    Persons associated with this session block
session_id
start_dt
timetable_entry	
title

class indico.modules.events.sessions.models.persons.SessionBlockPersonLink(*args, **kwargs)
    Bases: indico.modules.events.models.persons.PersonLinkBase
    Association between EventPerson and SessionBlock.
    Also known as a 'session convener'
display_order
id
object_relationship_name = u'session_block'
person
person_id
    person_link_backref_name = u'session_block_links'
    person_link_unique_columns = (u'session_block_id',)
    session_block_id

class indico.modules.events.sessions.models.principals.SessionPrincipal(**kwargs)
    Bases: indico.core.db.sqlalchemy.principals.PrincipalPermissionsMixin,
    sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.
    allow_category_roles = True
    allow_emails = True
    allow_event_roles = True
category_role
category_role_id
disallowed_protection_modes = frozenset([])
email
event_role
event_role_id
full_access

214 Chapter 5. API reference
id
   The ID of the acl entry
ip_network_group = None
ip_network_group_id = None
local_group
local_group_id
multipass_group_name
multipass_group_provider
permissions
principal_backref_name = u'in_session_acls'
principal_for = u'Session'
read_access
session_id
   The ID of the associated session
type
unique_columns = (u'session_id',)
user
user_id

Operations

indico.modules.events.sessions.operations.create_session(event, data)
   Create a new session with the information passed in the data argument
indico.modules.events.sessions.operations.create_session_block(session_, data)
indico.modules.events.sessions.operations.delete_session(event_session)
   Delete session from the event
indico.modules.events.sessions.operations.delete_session_block(session_block)
indico.modules.events.sessions.operations.update_session(event_session, data)
   Update a session based on the information in the data
indico.modules.events.sessions.operations.update_session_block(session_block, data)
   Update a session block with data passed in the data argument
indico.modules.events.sessions.operations.update_session_coordinator_privs(event, data)

Utilities

class indico.modules.events.sessions.util.SessionListToPDF(sessions)
   Bases: indico.legacy.pdfinterface.base.PDFBase
getBody (story=None)
indico.modules.events.sessions.util.can_manage_sessions(user, event, permission=None)

Check whether a user can manage any sessions in an event

indico.modules.events.sessions.util.generate_pdf_from_sessions(sessions)

Generate a PDF file from a given session list

indico.modules.events.sessions.util.generate_spreadsheet_from_sessions(sessions)

Generate spreadsheet data from a given session list.

Parameters
sessions – The sessions to include in the spreadsheet

indico.modules.events.sessions.util.get_events_with_linked_sessions(user, dt=None)

Returns a dict with keys representing event_id and the values containing data about the user rights for sessions within the event

Parameters
• user – A User
• dt – Only include events taking place on/after that date

indico.modules.events.sessions.util.get_session_ical_file(sess)

indico.modules.events.sessions.util.get_session_timetable_pdf(sess, **kwargs)

indico.modules.events.sessions.util.get_sessions_for_user(event, user)

indico.modules.events.sessions.util.has_sessions_for_user(event, user)

indico.modules.events.sessions.util.render_session_type_row(session_type)

indico.modules.events.sessions.util.serialize_session_for_ical(sess)

indico.modules.events.sessions.util.session_coordinator_priv_enabled(event, priv)

Check whether a coordinator privilege is enabled.
Currently the following privileges are available:
• manage-contributions
• manage-blocks

Parameters
• event – The Event to check for
• priv – The name of the privilege

5.1.17 Survey

Todo: Docstrings (module, models)

Models

class indico.modules.events.surveys.models.surveys.Survey(**kwargs)

Bases: sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in `kwargs`.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

**anonymous**
Whether submissions will not be linked to a user

**can_submit** *(user)*

**close**

**end_dt**
Datetime when the survey is closed

**event**
The Event containing this survey

**event_id**
The ID of the event

**has_ended**

**has_started**

**id**
The ID of the survey

**introduction**

**is_active**

**is_deleted**
Whether the survey has been marked as deleted

**is_visible**

**items**
The list of items

**limit_reached**

**locator**
Defines a smart locator property.
This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

5.1. API reference 217
**SurveyState**

- active_and_answered = 4
- active_and_clean = 3
- finished = 5
- limit_reached = 6
```python
default_render_mode = 2

display_as_section
    If a section should be rendered as a section

field_data
    Field-specific data (such as choices for multi-select fields)

field_type
    The type of the field used for the question

id
    The ID of the item

is_required
    If the question must be answered (wtforms DataRequired)

parent_id
    The ID of the parent section item (NULL for top-level items, i.e. sections)

position
    The position of the item in the survey form

possible_render_modes = set([<RenderMode.markdown: 2>])

render_mode = 2

survey_id
    The ID of the survey

title
    The title of the item

to_dict()
    Return a json-serializable representation of this object.
    Subclasses must add their own data to the dict.

type
    The type of the survey item
```

5.1. API reference
class indico.modules.events.surveys.models.items.SurveyQuestion(**kwargs)
    Bases: indico.modules.events.surveys.models.items.SurveyItem

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    display_as_section
    field
    field_data
    field_type
    get_summary(**kwargs)
        Returns the summary of answers submitted for this question.

    id
    is_required
    locator
    not_empty_answers
    parent_id
    position
    render_mode = 2
    survey_id
    title
    to_dict()
    type

class indico.modules.events.surveys.models.items.SurveySection(**kwargs)
    Bases: indico.modules.events.surveys.models.items.SurveyItem

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    children
        The child items of this section
    display_as_section
    field_data
    field_type
    id
    is_required
    locator
    parent_id
position
render_mode = 2
survey_id
title
to_dict()
type
class indico.modules.events.surveys.models.items.SurveyText(**kwargs)
Bases: indico.modules.events.surveys.models.items.SurveyItem

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
display_as_section
field_data
field_type
id
is_required
locator
parent_id
position
render_mode = 2
survey_id
title
to_dict()
type
class indico.modules.events.surveys.models.submissions.SurveyAnswer(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
answer_data
data
    The user’s answer (no, not 42!) to the question
is_empty
question
    The list of answers
question_id
   The ID of the question

submission_id
   The ID of the submission

class indico.modules.events.surveys.models.submissions.SurveySubmission(**kwargs)
   Bases: sqlalchemy.ext.declarative.api.Model
   A simple constructor that allows initialization from kwargs.
   Sets attributes on the constructed instance using the names and values in kwargs.
   Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
   mapped columns or relationships.

answers
   The list of answers

friendly_id
   The human-friendly ID of the submission

id
   The ID of the submission

is_anonymous
   Whether the survey submission is anonymous

is_submitted
   Whether the survey was submitted

locator

pending_answers
   List of non-submitted answers

submitted_dt
   The date/time when the survey was submitted

survey_id
   The ID of the survey

user
   The user who submitted the survey

user_id
   The ID of the user who submitted the survey

Operations

indico.modules.events.surveys.operations.add_survey_question(section, field_cls, data)
   Add a question to a survey.

   Parameters
       • section – The SurveySection to which the question will be added.
       • field_cls – The field class of this question.
       • data – The FieldConfigForm.data to populate the question with.

   Returns  The added SurveyQuestion.
indico.modules.events.surveys.operations.add_survey_section (survey, data)
Add a section to a survey.

Parameters

• **survey** – The Survey to which the section will be added.
• **data** – Attributes of the new SurveySection.

Returns The added SurveySection.

indico.modules.events.surveys.operations.add_survey_text (section, data)
Add a text item to a survey.

Parameters

• **section** – The SurveySection to which the question will be added.
• **data** – The TextForm.data to populate the question with.

Returns The added SurveyText.

Utilities

indico.modules.events.surveys.util.generate_spreadsheet_from_survey (survey, submission_ids)
Generates spreadsheet data from a given survey.

Parameters

• **survey** – Survey for which the user wants to export submissions
• **submission_ids** – The list of submissions to include in the file

indico.modules.events.surveys.util.get_events_with_submitted_surveys (user, dt=None)
Gets the IDs of events where the user submitted a survey.

Parameters

• **user** – A User
• **dt** – Only include events taking place on/after that date

Returns A set of event ids

indico.modules.events.surveys.util.is_submission_in_progress (survey)
Check whether the current user has a survey submission in progress

indico.modules.events.surveys.util.make_survey_form (survey)
Creates a WTForm from survey questions.

Parameters

• **survey** – The Survey for which to create the form.

Returns An IndicoForm subclass.

indico.modules.events.surveys.util.query_active_surveys (event)

indico.modules.events.surveys.util.save_submitted_survey_to_session (submission)
Save submission of a survey to session for further checks

indico.modules.events.surveys.util.was_survey_submitted (*args, **kwargs)
Check whether the current user has submitted a survey
## 5.1.18 Timetable

**Todo:** Docstring (module, models, operations, utilities)

### Models

```python
class indico.modules.events.timetable.models.breaks.Break(**kwargs):
    Bases: indico.core.db.sqlalchemy.descriptions.DescriptionMixin, indico.
core.db.sqlalchemy.colors.ColorMixin, indico.core.db.sqlalchemy.locations.
LocationMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwars.
Sets attributes on the constructed instance using the names and values in kwars.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.
```

- `background_color`
- `can_access(user)`
- `default_colors = ColorTuple(text=u'202020', background=u'90c0f0')`
- `default_render_mode = 2`
- `duration`
- `end_dt`
- `event`
- `id`
- `inherit_location`
- `location_backref_name = u'breaks'`
- `location_parent`
- `locator`
  Defines a smart locator property.
  This behaves pretty much like a normal read-only property and the decorated function should return a dict
  containing the necessary data to build a URL for the object.
  This decorator should usually be applied to a method named `locator` as this name is required for
  `get_locator` to find it automatically when just passing the object.
  If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an
object with a locator.
own_address
own_room
own_room_id
own_room_name
own_venue
own_venue_id
own_venue_name
possible_render_modes = set([<RenderMode.markdown: 2>])
render_mode = 2
start_dt
text_color
timetable_entry
title

class indico.modules.events.timetable.models.entries.TimetableEntry(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

break_
break_id
can_view(user)
    Checks whether the user will see this entry in the timetable.
children
contribution
contribution_id
duration
end_dt
event
event_id
extend_end_dt(end_dt)
extend_parent(by_start=True, by_end=True)
    Extend start/end of parent objects if needed.
    No extension if performed for entries crossing a day boundary in the event timezone.

Parameters
• by_start – Extend parent by start datetimetime.
• by_end – Extend parent by end datetimetime.

5.1. API reference
extend_start_dt (start_dt)

id

is_parallel (in_session=False)

locator
  Defines a smart locator property.

  This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

  This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

  If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.

move (start_dt)

  Move the entry to start at a different time.

  This method automatically moves children of the entry to preserve their start time relative to the parent’s start time.

move_next_to (sibling, position=u'before')

object

parent

parent_id

session_block

session_block_id

session_siblings

siblings

siblings_query

start_dt

type

class indico.modules.events.timetable.models.entries.TimetableEntryType
  Bases: indico.util.struct.enum.RichIntEnum

  BREAK = 3
  CONTRIBUTION = 2
  SESSION_BLOCK = 1
Operations

```
indico.modules.events.timetable.operations.can_swap_entry(entry, direction, in_session=False)
indico.modules.events.timetable.operations.create_break_entry(event, data, session_block=None)
indico.modules.events.timetable.operations.create_session_block_entry(session_, data)
indico.modules.events.timetable.operations.create_timetable_entry(event, data, parent=None, extend_parent=False)
indico.modules.events.timetable.operations.delete_timetable_entry(entry, log=True)
indico.modules.events.timetable.operations.fit_session_block_entry(entry, log=True)
indico.modules.events.timetable.operations.get_sibling_entry(entry, direction, in_session=False)
indico.modules.events.timetable.operations.move_timetable_entry(entry, parent=None, day=None)
```

Move the entry to another session or top-level timetable

Parameters

- `entry` – TimetableEntry to be moved
- `parent` – If specified then the entry will be set as a child of parent
- `day` – If specified then the entry will be moved to the top-level timetable on this day

```
indico.modules.events.timetable.operations.schedule_contribution(contribution, start_dt, session_block=None, extend_parent=False)
indico.modules.events.timetable.operations.swap_timetable_entry(entry, direction, session_=None)
```

Swap entry with closest gap or non-parallel sibling

```
indico.modules.events.timetable.operations.update_break_entry(break_, data)
indico.modules.events.timetable.operations.update_timetable_entry(entry, data)
indico.modules.events.timetable.operations.update_timetable_entry_object(entry, data)
```

Update the object of a timetable entry according to its type

Utilities

```
indico.modules.events.timetable.util.find_latest_entry_end_dt(obj, day=None)
```

Get the latest end datetime for timetable entries within the object.
Parameters

- **obj** – The Event or SessionBlock that will be used to look for timetable entries.
- **day** – The local event date to look for timetable entries. Applicable only to Event.

Returns

The end datetime of the timetable entry finishing the latest. None if no entry was found.

```python
indico.modules.events.timetable.util.find_next_start_dt(duration, obj, day=None, force=False)
```

Find the next most convenient start date fitting a duration within an object.

Parameters

- **duration** – Duration to fit into the event/session-block.
- **obj** – The Event or SessionBlock the duration needs to fit into.
- **day** – The local event date where to fit the duration in case the object is an event.
- **force** – Gives earliest datetime if the duration doesn’t fit.

Returns

The end datetime of the latest scheduled entry in the object if the duration fits then. It it
doesn’t, the latest datetime that fits it. None if the duration cannot fit in the object, earliest
datetime if **force** is True.

```python
indico.modules.events.timetable.util.get_category_timetable(categ_ids, start_dt, end_dt, detail_level=u'event', tz=<UTC>, from_categ=None, grouped=True)
```

Retrieve time blocks that fall within a specific time interval for a given set of categories.

Parameters

- **categ_ids** – iterable containing list of category IDs
- **start_dt** – start of search interval (datetime, expected to be in display timezone)
- **end_dt** – end of search interval (datetime in expected to be in display timezone)
- **detail_level** – the level of detail of information (event|session|contribution)
- **tz** – the timezone information should be displayed in
- **from_categ** – Category that will be taken into account to calculate visibility
- **grouped** – Whether to group results by start date

Returns

a dictionary containing timetable information in a structured way. See source code for
examples.

```python
indico.modules.events.timetable.util.get_nested_entries(*args, **kwargs)
indico.modules.events.timetable.util.get_session_block_entries(event, day)
```

Returns a list of event top-level session blocks for the given day

```python
indico.modules.events.timetable.util.get_time_changes_notifications(changes, tzinfo, entry=None)
```

```python
indico.modules.events.timetable.util.get_timetable_offline_pdf_generator(event)
```

```python
indico.modules.events.timetable.util.get_top_level_entries(*args, **kwargs)
```
indico.modules.events.timetable.util.render_entry_info_balloon(entry, editable=False, sess=None, is_session_timetable=False)

indico.modules.events.timetable.util.render_session_timetable(session, timetable_layout=None, management=False)

indico.modules.events.timetable.util.shift_following_entries(entry, shift, session_=None)

Reschedules entries starting after the given entry by the given shift.

class indico.modules.events.timetable.reschedule.RescheduleMode
    Bases: unicode, indico.util.struct.enum.RichEnum

duration = u'duration'
none = u'none'
time = u'time'

class indico.modules.events.timetable.reschedule.Rescheduler(event, mode, day, session=None, session_block=None, fit_blocks=False, gap=datetime.timedelta(0))

    Bases: object

    Compacts the the schedule of an event day by either adjusting start times or durations of timetable entries.

    Parameters

    • **event** – The event of which the timetable entries should be rescheduled.
    • **mode** – A RescheduleMode value specifying whether the duration or start time should be adjusted.
    • **day** – A date specifying the day to reschedule (the day of the timetable entries are determined using the event’s timezone)
    • **session** – If specified, only blocks of that session will be rescheduled, ignoring any other timetable entries. Cannot be combined with session_block.
    • **session_block** – If specified, only entries inside that block will be rescheduled. Cannot be combined with session.
    • **fit_blocks** – Whether session blocks should be resized to exactly fit their contents before the actual rescheduling operation.
    • **gap** – A timedelta specifying the cap between rescheduled timetable entries.

    run()
        Perform the rescheduling

5.1.19 Track

Todo: Docstring (module, models, operations)
Models

class indico.modules.events.tracks.models.tracks.Track(**kwargs)

Bases: indico.core.db.sqlalchemy.descriptions.DescriptionMixin, indico.core.db.sqlalchemy.protection.ProtectionManagersMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

access_key = None

acl_entries

can_convene(user)

can_delete(user)

can_review_abstracts(user)

code

default_render_mode = 2

default_session

default_session_id

disable_protection_mode = True

event

event_id

full_title

full_title_with_group

id

is_track_group = False

locator

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```
The other locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
own_no_access_contact = None
position
possible_render_modes = set([<RenderMode.markdown: 2>])
protection_mode = None
render_mode = 2
short_title
short_title_with_group
title
title_with_group
track_group
track_group_id
indico.modules.events.tracks.models.tracks.get_next_position(context)
```

```python
class indico.modules.events.tracks.models.principals.TrackPrincipal(**kwargs)
    Bases: indico.core.db.sqlalchemy.principals.PrincipalPermissionsMixin,
           sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    allow_category_roles = True
    allow_emails = True
    allow_event_roles = True
    category_role
    category_role_id
    email
    event_role
    event_role_id
    full_access
    id
    ip_network_group = None
    ip_network_group_id = None
    local_group
    local_group_id
    multipass_group_name
    multipass_group_provider
```

5.1. API reference
permissions
principal_backref_name = u'in_track_acls'
principal_for = u'Track'
read_access
track_id
type
unique_columns = (u'track_id',)
user
user_id

Operations
indico.modules.events.tracks.operations.create_track(event, data)
indico.modules.events.tracks.operations.create_track_group(event, data)
indico.modules.events.tracks.operations.delete_track(track)
indico.modules.events.tracks.operations.delete_track_group(track_group)
indico.modules.events.tracks.operations.update_program(event, data)
indico.modules.events.tracks.operations.update_track(track, data)
indico.modules.events.tracks.operations.update_track_group(track_group, data)

5.1.20 Static site

Todo: Doctrings (module, utilities)

Models
class indico.modules.events.static.models.static.StaticSite(**kwargs)
    Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model

    Static site for an Indico event.
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    add_file_date_column = False
    content_type
        The MIME type of the file
    created_dt = None
creator
The user who created the static site

creator_id
ID of the user who created the static site

event
The Event this static site is associated with

event_id
ID of the event

file_required = False

filename
The name of the file

id
Entry ID

locator

md5
An MD5 hash of the file.
Automatically assigned when save() is called.

requested_dt
The date and time the static site was requested

size
The size of the file (in bytes).
Automatically assigned when save() is called.

state
The state of the static site (a StaticSiteState member)

storage_backend

storage_file_id

class indico.modules.events.static.models.static.StaticSiteState
Bases: indico.util.struct.enum.RichIntEnum

expired = 4
failed = 3
pending = 0
running = 1
success = 2

Utilities

class indico.modules.events.static.util.RewrittenManifest (manifest)
Bases: pywebpack.manifests.Manifest

A manifest that rewrites its asset paths.

indico.modules.events.static.util.collect_static_files (*args, **kwds)
Keep track of URLs used by manifest and url_for.
indico.modules.events.static.util.override_request_endpoint(*args, **kwds)

Rewrite CSS in order to handle url(...) properly.

indico.modules.events.static.util.rewrite_css_urls(event, css)

Removing __vxxx prefix from static URLs.

indico.modules.events.static.util.rewrite_static_url(path)

Handle special endpoint/URLs so that they link to offline content.

5.1.21 Category

Todo: Docstrings (module, model, operations, utilities)

Models

class indico.modules.categories.models.categories.Category(**kwargs)

Bases: indico.core.db.sqlalchemy.searchable_titles.SearchableTitleMixin, indico.core.db.sqlalchemy.descriptions.DescriptionMixin, indico.core.db.sqlalchemy.protection.ProtectionManagersMixin, indico.core.db.sqlalchemy.attachments.AttachedItemsMixin, sqlalchemy.ext.declarative.api.Model

An Indico category

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

ATTACHMENT_FOLDER_ID_COLUMN = u'category_id'

access_key = None

acl_entries

allow_no_access_contact = True

can_create_events(user)

Check whether the user can create events in the category.

chain_query

Get a query object for the category chain.

The query retrieves the root category first and then all the intermediate categories up to (and including) this category.

children

depth_childrren_query

Get a query object for all subcategories.

This includes subcategories at any level of nesting.

default_event_themes

default_render_mode = 2

234 Chapter 5. API reference
default_ticket_template
default_ticket_template_id
disallowed_protection_modes = frozenset([])
display_tzinfo
    The tzinfo of the category or the one specified by the user
effective_icon_url
    Get the HTTP URL of the icon (possibly inherited).
event_creation_notification_emails
event_creation_restricted
event_message
event_message_mode
classmethod get_icon_data_cte()
classmethod get_protection_cte()
get_protection_parent_cte()
classmethod get_root()
    Get the root category
classmethod get_tree_cte(col=u'id')
    Create a CTE for the category tree.
    The CTE contains the following columns:
    • id – the category id
    • path – an array containing the path from the root to the category itself
    • is_deleted – whether the category is deleted

    Parameters col – The name of the column to use in the path or a callable receiving the category
    alias that must return the expression used for the ‘path’ retrieved by the CTE.
static get_visible_categories_cte(category_id)
    Get a sqlalchemy select for the visible categories within the given category, including the category itself.
has_effective_icon
has_icon
has_logo
has_only_events
icon
icon_metadata
icon_url
    Get the HTTP URL of the icon.
id
inheriting_have_acl = True
is_deleted
is_descendant_of(categ)
is_empty
is_root
legacy_mapping
locator
   Defines a smart locator property.
   This behaves pretty much like a normal read-only property and the decorated function should return a dict
   containing the necessary data to build a URL for the object.
   This decorator should usually be applied to a method named locator as this name is required for
   get_locator to find it automatically when just passing the object.
   If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator.other
def locator(self):
    return {...}
```

The other locator can then be accessed by passing obj.locator.other to the code expecting an
object with a locator.

logo

logo_metadata

logo_url
   Get the HTTP URL of the logo.

move ( target )
   Move the category into another category.

notify_managers

nth_parent ( n_categs, fail_on_overflow=True )
   Return the nth parent of the category.

   Parameters
   • n_categs – the number of categories to go up
   • fail_on_overflow – whether to fail if we try to go above the root category

   Returns Category object or None (only if fail_on_overflow is not set)

own_no_access_contact

own_visibility_horizon
   Get the highest category this one would like to be visible from (configured visibility).

parent

parent_chain_query
   Get a query object for the category’s parent chain.
   The query retrieves the root category first and then all the intermediate categories up to (excluding) this
category.

parent_id
position
possible_render_modes = set([ RenderMode.markdown: 2 ])
protection_mode
protection_parent
real_visibility_horizon
Get the highest category this one is actually visible from (as limited by categories above).
render_mode = 2
suggestions_disabled
timezone
title
tzinfo
url
visibility
visibility_horizon_query
Get a query object that returns the highest category this one is visible from.
visible_categories_query
Get a query object for the visible categories within this category, including the category itself.

class indico.modules.categories.models.categories.EventMessageMode
Bases: indico.util.struct.enum.RichIntEnum
danger = 3
disabled = 0
info = 1
warning = 2

class indico.modules.categories.models.principals.CategoryPrincipal(**kwargs)
Bases: indico.core.db.sqlalchemy.principals.PrincipalPermissionsMixin,
        sqlalchemy.ext.declarative.api.Model
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.
allow_category_roles = True
allow_networks = True
category
category_id
    The ID of the associated event
category_role
category_role_id
event_role = None
event_role = None
event_role_id = None
full_access
id
    The ID of the acl entry
ip_network_group
ip_network_group_id
local_group
local_group_id
multipass_group_name
multipass_group_provider
permissions
principal_backref_name = u'in_category_acls'
principal_for = u'Category'
read_access
type
unique_columns = (u'category_id',)
user
user_id
class indico.modules.categories.models.settings.CategorySetting(**kwargs)
    Bases: indico.core.settings.models.base.JSONSettingsBase, sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.
    category
category_id
    id
module
name
value

Operations

indico.modules.categories.operations.create_category(parent, data)
indico.modules.categories.operations.delete_category(category)
indico.modules.categories.operations.move_category(category, target_category)
indico.modules.categories.operations.update_category(category, data, skip=())
Utilities

```python
indico.modules.categories.util.get_attachment_count(category_id=None)
```
Get the number of attachments in events in a category.

**Parameters**
- `category_id` – The category ID to get statistics for. Attachments from subcategories are also included.

**Returns**
The number of attachments

```python
indico.modules.categories.util.get_category_stats(*args, **kwargs)
```
Get category statistics.

This function is mainly a helper so we can get and cache all values at once and keep a last-update timestamp.

**Parameters**
- `category_id` – The category ID to get statistics for. Subcategories are also included.

```python
indico.modules.categories.util.get_contribs_by_year(category_id=None)
```
Get the number of contributions for each year.

**Parameters**
- `category_id` – The category ID to get statistics for. Contributions from subcategories are also included.

**Returns**
An `OrderedDict` mapping years to contribution counts.

```python
indico.modules.categories.util.get_events_by_year(category_id=None)
```
Get the number of events for each year.

**Parameters**
- `category_id` – The category ID to get statistics for. Events from subcategories are also included.

**Returns**
An `OrderedDict` mapping years to event counts.

```python
indico.modules.categories.util.get_image_data(image_type, category)
```
Get image data

```python
indico.modules.categories.util.get_upcoming_events(*args, **kwargs)
```
Get the global list of upcoming events

```python
indico.modules.categories.util.get_visibility_options(category_or_event, allow_invisible=True)
```
Return the visibility options available for the category or event.

```python
indico.modules.categories.util.serialize_category_role(role, legacy=True)
```
Serialize role to JSON-like object

```python
indico.modules.categories.serialize.serialize_categories_ical(category_ids, user, event_filter=True, event_filter_fn=None, update_query=None)
```
Export the events in a category to iCal

**Parameters**
- `category_ids` – Category IDs to export
- `user` – The user who needs to be able to access the events
- `event_filter` – A SQLAlchemy criterion to restrict which events will be returned. Usually something involving the start/end date of the event.
- `event_filter_fn` – A callable that determines which events to include (after querying)
• **update_query** – A callable that can update the query used to retrieve the events. Must return the updated query object.

```python
indico.modules.categories.serialize.serialize_category(category,
    with_favorite=False,
    with_path=False,
    parent_path=None,
    child_path=None)
```

```python
indico.modules.categories.serialize.serialize_category_atom(category, url, user,
    event_filter)
```

Export the events in a category to Atom

**Parameters**

- **category** – The category to export
- **url** – The URL of the feed
- **user** – The user who needs to be able to access the events
- **event_filter** – A SQLAlchemy criterion to restrict which events will be returned. Usually something involving the start/end date of the event.

```python
indico.modules.categories.serialize.serialize_category_chain(category, in-
    clude_children=False,
    include_parents=False)
```

**Settings**

```python
class indico.modules.categories.settings.CategorySettingsProxy(module,
    defaults=None,
    strict=True,
    acls=None,
    converters=None)
```

**Bases:** `indico.core.settings.proxy.SettingsProxyBase`

Proxy class to access category-specific settings for a certain module.

```python
delete(category, *args, **kwargs)
```

Delete settings.

**Parameters**

- **category** – Category (or its ID)
- **names** – One or more names of settings to delete

```python
delete_all(category, *args, **kwargs)
```

Delete all settings.

**Parameters**

- **category** – Category (or its ID)

```python
get(category, *args, **kwargs)
```

Retrieve the value of a single setting.

**Parameters**

- **category** – Category (or its ID)
- **name** – Setting name
- **default** – Default value in case the setting does not exist
Returns The settings’s value or the default value

**get_all** *(category, *args, **kwargs)*
Retrieve all settings.

Parameters

- **category** – Category (or its ID)
- **no_defaults** – Only return existing settings and ignore defaults.

Returns Dict containing the settings

**query**
Return a query object filtering by the proxy’s module.

**set** *(category, *args, **kwargs)*
Set a single setting.

Parameters

- **category** – Category (or its ID)
- **name** – Setting name
- **value** – Setting value; must be JSON-serializable

**set_multi** *(category, *args, **kwargs)*
Set multiple settings at once.

Parameters

- **category** – Category (or its ID)
- **items** – Dict containing the new settings

### 5.1.22 User

**Todo:** Docstrings (module, models, utilities)

**Models**

```python
class indico.modules.users.models.users.NameFormat
    Bases: indico.util.struct.enum.RichIntEnum
    f_last = 3
    f_last_upper = 7
    first_last = 0
    first_last_upper = 4
    last_f = 2
    last_f_upper = 6
    last_first = 1
    last_first_upper = 5
```

5.1. API reference 241
class indico.modules.users.models.users.PersonMixin
    Bases: object

    Add convenience properties and methods to person classes.
    Assumes the following attributes exist: * first_name * last_name * title

    display_full_name
      Return the full name using the user’s preferred name format.

    full_name
      Return the person’s name in ‘Firstname Lastname’ notation.

    get_full_name(last_name_first=True, last_name_upper=True, abbrev_first_name=True, show_title=False, _show_empty_names=False)
      Return the person’s name in the specified notation.
      Note: Do not use positional arguments when calling this method. Always use keyword arguments!

      Parameters
        • last_name_first – if “lastname, firstname” instead of “firstname lastname” should be used
        • last_name_upper – if the last name should be all-uppercase
        • abbrev_first_name – if the first name should be abbreviated to use only the first character
        • show_title – if the title of the person should be included

    name
      Return the person’s name in ‘Firstname Lastname’ notation.

    title
      The title of the user

class indico.modules.users.models.users.User(**kwargs)
    Bases: indico.modules.users.models.users.PersonMixin, sqlalchemy.ext.declarative.api.Model

    Indico users

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in kwargs.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    address
      the address of the user

    affiliation
      the affiliation of the user

    all_emails
      all emails of the user. read-only; use it only for searching by email! also, do not use it between modifying email or secondary_emails and a session expire/commit!

    api_key
      the active API key of the user

    as_avatar
    as_legacy
as_principal
    The serializable principal identifier of this user

avatar_bg_color

avatar_css

can_be_modified(user)
    If this user can be modified by the given user

can_get_all_multipass_groups
    Check whether it is possible to get all multipass groups the user is in.

email
    the primary email address of the user

event_log_entries

event_notes_revisions

external_identities
    The external identities of the user

favorite_categories
    the users’s favorite categories

favorite_users
    the users’s favorite users

first_name
    the first name of the user

get_full_name(*args, **kwargs)

static get_system_user()

id
    the unique id of the user

identifier

identities
    the identities used by this user

in_attachment_folder_acls

in_room_acls

is_admin
    if the user is an administrator with unrestricted access to everything

is_blocked
    if the user has been blocked

is_category_role = False

is_deleted
    if the user is deleted (e.g. due to a merge)

is_event_role = False

is_group = False

is_network = False

is_pending
    if the user is pending (e.g. never logged in, only added to some list)
is_single_person = True

is_system
    if the user is the default system user

iter_all_multipass_groups()
    Iterate over all multipass groups the user is in

iter_identifiers (check_providers=False, providers=None)
    Yields (provider, identifier) tuples for the user.

Parameters

• check_providers – If True, providers are searched for additional identifiers once all
  existing identifiers have been yielded.

• providers – May be a set containing provider names to get only identifiers from the
  specified providers.

last_name
    the last/family name of the user

local_identities
    The local identities of the user

local_identity
    The main (most recently used) local identity

locator
    Defines a smart locator property.

    This behaves pretty much like a normal read-only property and the decorated function should return a dict
    containing the necessary data to build a URL for the object.

    This decorator should usually be applied to a method named locator as this name is required for
    get_locator to find it automatically when just passing the object.

    If you need more than one locator, you can define it like this:

    @locator_property
def locator(self):
        return {...}

    @locator.other
def locator(self):
        return {...}

    The other locator can then be accessed by passing obj.locator.other to the code expecting an
    object with a locator.

make_email_primary (email)
    Promotes a secondary email address to the primary email address

    Parameters email – an email address that is currently a secondary email

merged_into_id
    the id of the user this user has been merged into

merged_into_user
    the user this user has been merged into

old_api_keys
    the previous API keys of the user
phone
    the phone number of the user
principal_order = 0
principal_type = 1
reservations
reservations_booked_for
reset_signing_secret()
secondary_emails
    any additional emails the user might have
secondary_local_identities
    The local identities of the user except the main one
settings
    Returns the user settings proxy for this user
signing_secret
    a unique secret used to generate signed URLs
suggested_categories
    the user’s category suggestions
synced_fields
    The fields of the user whose values are currently synced.
    This set is always a subset of the synced fields define in synced fields of the idp in ‘indico.conf’.
synced_values
    The values from the synced identity for the user.
    Those values are not the actual user’s values and might differ if they are not set as synchronized.
synchronize_data(refresh=False)
    Synchronize the fields of the user from the sync identity.
    This will take only into account synced_fields.

Parameters refresh – bool – Whether to refresh the synced identity with the sync provider before instead of using the stored data. (Only if the sync provider supports refresh.)

class indico.modules.users.models.users.UserTitle
    Bases: indico.util.struct.enum.RichIntEnum
    dr = 4
    mr = 1
    mrs = 3
    ms = 2
    none = 0
    prof = 5
indico.modules.users.models.users.format_display_full_name(user, obj)
indico.modules.users.models.users.syncable_fields = {u'address': lu'address', u'affiliation'
    Fields which can be synced as keys and a mapping to a more human readable version, used for flashing messages
class indico.modules.users.models.affiliations.UserAffiliation(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    
    A simple constructor that allows initialization from kwargs.
    
    Sets attributes on the constructed instance using the names and values in kwargs.
    
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.

    id
        the unique id of the affiliations

    name
        the affiliation

    user_id
        the id of the associated user

class indico.modules.users.models.emails.UserEmail(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    
    A simple constructor that allows initialization from kwargs.
    
    Sets attributes on the constructed instance using the names and values in kwargs.
    
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.

    email
        the email address

    id
        the unique id of the email address

    is_primary
        if the email is the user’s primary email

    is_user_deleted
        if the user is marked as deleted (e.g. due to a merge). DO NOT use this flag when actually deleting an
        email

    user_id
        the id of the associated user

class indico.modules.users.models.suggestions.SuggestedCategory(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    
    A simple constructor that allows initialization from kwargs.
    
    Sets attributes on the constructed instance using the names and values in kwargs.
    
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
    mapped columns or relationships.

    category

    category_id

    is_ignored

    classmethod merge_users(target, source)
        Merge the suggestions for two users.

        Parameters
• **target** – The target user of the merge.
• **source** – The user that is being merged into **target**.

code

```python
score
user_id
```

class indico.modules.users.models.settings.UserSetting(**kwargs)
Bases: indico.core.settings.models.base.JSONSettingsBase, sqlalchemy.ext.declarative.api.Model

User-specific settings
A simple constructor that allows initialization from **kwargs**.
Sets attributes on the constructed instance using the names and values in **kwargs**.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

code

```python
id
module
name
user
user_id
value
```

class indico.modules.users.models.settings.UserSettingsProxy(module, defaults=None, strict=True, acls=None, converters=None)
Bases: indico.core.settings.proxy.SettingsProxyBase

Proxy class to access user-specific settings for a certain module

def delete(user, *args, **kwargs)
    Deletes settings.
    Parameters
    • **user** – {'user': user} or {'user_id': id}
    • **names** – One or more names of settings to delete

def delete_all(user, *args, **kwargs)
    Deletes all settings.
    Parameters **user** – {'user': user} or {'user_id': id}

def get(user, *args, **kwargs)
    Retrieves the value of a single setting.
    Parameters
    • **user** – {'user': user} or {'user_id': id}
    • **name** – Setting name
    • **default** – Default value in case the setting does not exist
    Returns The setting’s value or the default value

5.1. API reference 247
get_all (user, *args, **kwargs)
Retrieves all settings

Parameters
• **user** – {'user': user} or {'user_id': id}
• **no_defaults** – Only return existing settings and ignore defaults.

Returns  Dict containing the settings

query
Returns a query object filtering by the proxy’s module.

set (user, *args, **kwargs)
Sets a single setting.

Parameters
• **user** – {'user': user} or {'user_id': id}
• **name** – Setting name
• **value** – Setting value; must be JSON-serializable

set_multi (user, *args, **kwargs)
Sets multiple settings at once.

Parameters
• **user** – {'user': user} or {'user_id': id}
• **items** – Dict containing the new settings

indico.modules.users.models.settings.user_or_id (f)

Operations

indico.modules.users.operations.create_user (email, data, identity=None, settings=None, other_emails=None, from_moderation=True)
Create a new user.

This may also convert a pending user to a proper user in case the email address matches such a user.

Parameters
• **email** – The primary email address of the user.
• **data** – The data used to populate the user.
• **identity** – An Identity to associate with the user.
• **settings** – A dict containing user settings.
• **other_emails** – A set of email addresses that are also used to check for a pending user. They will also be added as secondary emails to the user.
• **from_moderation** – Whether the user was created through the moderation process or manually by an admin.
Utilities

```python
indico.modules.users.util.build_user_search_query(criteria, exact=False, include_deleted=False, include_pending=False, favorites_first=False)
```

```python
indico.modules.users.util.get_admin_emails()
```
Get the email addresses of all Indico admins

```python
indico.modules.users.util.get_color_for_username(username)
```

```python
indico.modules.users.util.get_linked_events(user, dt, limit=None, load_also=())
```
Get the linked events and the user’s roles in them

**Parameters**

- **user** – A `User`
- **dt** – Only include events taking place on/after that date
- **limit** – Max number of events

```python
indico.modules.users.util.get_related_categories(user, detailed=True)
```
Gets the related categories of a user for the dashboard

```python
indico.modules.users.util.get_suggested_categories(user)
```
Gets the suggested categories of a user for the dashboard

```python
indico.modules.users.util.get_user_by_email(email, create_pending=False)
```
finds a user based on his email address.

**Parameters**

- **email** – The email address of the user.
- **create_pending** – If True, this function searches for external users and creates a new pending User in case no existing user was found.

**Returns**  
A `User` instance or `None` if not exactly one user was found.

```python
indico.modules.users.util.merge_users(source, target, force=False)
```
Merge two users together, unifying all related data

**Parameters**

- **source** – source user (will be set as deleted)
- **target** – target user (final)

```python
indico.modules.users.util.search_users(exact=False, include_deleted=False, include_pending=False, external=False, allow_system_user=False, **criteria)
```
Searches for users.

**Parameters**

- **exact** – Indicates if only exact matches should be returned. This is MUCH faster than a non-exact search, especially when searching external users.
- **include_deleted** – Indicates if also users marked as deleted should be returned.
- **include_pending** – Indicates if also users who are still pending should be returned.
- **external** – Indicates if identity providers should be searched for matching users.
- **allow_system_user** – Whether the system user may be returned in the search results.
• **criteria** – A dict containing any of the following keys: name, first_name, last_name, email, affiliation, phone, address

**Returns** A set of matching users. If *external* was set, it may contain both *IdentityInfo* objects for external users not yet in Indico and *User* objects for existing users.

```python
indico.modules.users.util.serialize_user(user)
```

Serialize user to JSON-like object

```python
class indico.modules.users.ext.ExtraUserPreferences(user)
```

Bases: object

Defines additional user preferences

To use this class, subclass it and override *defaults, fields* and *save* to implement your custom logic.

```python
extend_defaults(defaults)
```

Adds values to the FormDefaults.

```python
extend_form(form_class)
```

Create a subclass of the form containing the extra field

```python
fields = {}
```

a dict containing all the fields that should be added to the user preferences

```python
classmethod is_active(user)
```

Return whether the preferences are available for the given user.

```python
load()
```

Return a dict with the current values for the user.

```python
process_form_data(data)
```

Process and save submitted data.

This modifies *data* so the core code doesn’t receive any extra data it doesn’t expect.

```python
save(data)
```

Save the updated settings.

### 5.1.23 Attachment

**Todo:** Docstrings (module, models, operations)

#### Models

```python
class indico.modules.attachments.models.attachments.Attachment(**kwargs)
```


A simple constructor that allows initialization from *kwargs*.

Sets attributes on the constructed instance using the names and values in *kwargs*.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
absolute_download_url
```

The absolute download url for the attachment
access_key = None

acl
The ACL of the folder (used for ProtectionMode.protected)

acl_entries

all_files

can_access(user, *args, **kwargs)
Checks if the user is allowed to access the attachment.
This is the case if the user has access to see the attachment or if the user can manage attachments for the linked object.

description
The description of the attachment

download_url
The download url for the attachment

file

file_id

folder
The folder containing the attachment

folder_id
The ID of the folder the attachment belongs to

get_download_url(absolute=False)
Returns the download url for the attachment.
During static site generation this returns a local URL for the file or the target URL for the link.

Parameters

absolute – If the returned URL should be absolute.

id
The ID of the attachment

is_deleted
If the attachment has been deleted

link_url
The target URL for a link attachment

locator

modified_dt
The date/time when the attachment was created/modified

own_no_access_contact = None

protection_mode

protection_parent

stored_file_class
alias of AttachmentFile

stored_file_fkey = u'attachment_id'

stored_file_table = u'attachments.files'

title
The name of the attachment

5.1. API reference
type
   The type of the attachment (file or link)
user
   The user who created the attachment
user_id
   The ID of the user who created the attachment

```python
class AttachmentFile(**kwargs)
   Bases: indico.core.storage.models.StoredFileMixin,
          sqlalchemy.ext.declarative.api.Model
```
A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in `kwargs`.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
attachment_id
   The ID of the associated attachment
content_type
   The MIME type of the file
created_dt
   The date/time when the file was uploaded
filename
   The name of the file
id
   The ID of the file
is_previewable
md5
   An MD5 hash of the file.
   Automatically assigned when `save()` is called.
size
   The size of the file (in bytes).
   Automatically assigned when `save()` is called.
storage_backend
storage_file_id
user
   The user who uploaded the file
user_id
   The user who uploaded the file
version_of = u'attachment'
```
class indico.modules.attachments.models.folders.AttachmentFolder(**kwargs)

Bases: indico.core.db.sqlalchemy.links.LinkMixin, indico.core.db.sqlalchemy.protection.ProtectionMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

access_key = None

acl
    The ACL of the folder (used for ProtectionMode.protected)

acl_entries

allowed_link_types = frozenset([<LinkType.category: 1>, <LinkType.event: 2>, <LinkType.contribution: 3>, <LinkType.subcontribution: 4>, <LinkType.session: 5>])

attachments
    The list of attachments that are not deleted, ordered by name

can_access(user, *args, **kwargs)
    Checks if the user is allowed to access the folder.

    This is the case if the user has access the folder or if the user can manage attachments for the linked object.

can_view(user)
    Checks if the user can see the folder.

    This does not mean the user can actually access its contents. It just determines if it is visible to him or not.

category

category_id

contribution

contribution_id

description
    The description of the folder

event

event_id

events_backref_name = u'all_attachment_folders'

classmethod get_for_linked_object(linked_object, preload_event=False)
    Gets the attachments for the given object.

    This only returns attachments that haven’t been deleted.

    Parameters
        - linked_object: A category, event, session, contribution or subcontribution.
        - preload_event: If all attachments for the same event should be pre-loaded and cached in the app context. This must not be used when linked_object is a category.

classmethod get_or_create(linked_object, title=None)
    Gets a folder for the given object or creates it.
If no folder title is specified, the default folder will be used. It is the caller’s responsibility to add the folder or an object (such as an attachment) associated with it to the SQLAlchemy session using `db.session.add(...)`.

```python
classmethod get_or_create_default(linked_object)
    Gets the default folder for the given object or creates it.
```

- **id**
  - The ID of the folder

- **is_always_visible**
  - If the folder is always visible (even if you cannot access it)

- **is_default**
  - If the folder is the default folder (used for “folder-less” files)

- **is_deleted**
  - If the folder has been deleted

- **is_hidden**
  - If the folder is never shown in the frontend (even if you can access it)

- **link_backref_lazy = u'dynamic'**
- **link_backref_name = u'attachment_folders'**

- **link_type**

- **linked_event**

- **linked_event_id**

- **locator**
  - Defines a smart locator property.

  This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

  This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

  If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def other(locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

- **own_no_access_contact = None**

- **protection_mode**

- **protection_parent**

- **session**

- **session_block = None**

- **session_block_id = None**
session_id
subcontribution
subcontribution_id
title
   The name of the folder (None for the default folder)
unique_links = u'is_default'

class indico.modules.attachments.models.principals.AttachmentFolderPrincipal(**kwargs)
   Bases: indico.core.db.sqlalchemy.principals.PrincipalMixin, sqlalchemy.ext.declarative.api.Model
   A simple constructor that allows initialization from kwargs.
   Sets attributes on the constructed instance using the names and values in kwargs.
   Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
   allow_category_roles = True
   allow_event_roles = True
category_role
category_role_id
event_role
event_role_id
folder_id
   The ID of the associated folder
id
   The ID of the acl entry
ip_network_group = None
ip_network_group_id = None
local_group
local_group_id
multipass_group_name
multipass_group_provider
principal_backref_name = u'in_attachment_folder_acls'
type
unique_columns = (u'folder_id',)
user
user_id

class indico.modules.attachments.models.principals.AttachmentPrincipal(**kwargs)
   Bases: indico.core.db.sqlalchemy.principals.PrincipalMixin, sqlalchemy.ext.declarative.api.Model
   A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
allow_category_roles = True
allow_event_roles = True
attachment_id
    The ID of the associated attachment
category_role
category_role_id
event_role
event_role_id
id
    The ID of the acl entry
ip_network_group = None
ip_network_group_id = None
local_group
local_group_id
multipass_group_name
multipass_group_provider
principal_backref_name = u'\in_attachment_acls'
type
unique_columns = (u'attachment_id',)
user
user_id
```

**Operations**

```python
indico.modules.attachments.operations.add_attachment_link(data, linked_object)
```

Add a link attachment to linked_object

**Utilities**

```python
indico.modules.attachments.util.can_manage_attachments(obj, user)
```

Checks if a user can manage attachments for the object

```python
indico.modules.attachments.util.get_attached_folders(linked_object, include_empty=True, include_hidden=True, preload_event=False)
```

Return a list of all the folders linked to an object.

**Parameters**
• **linked_object** – The object whose attachments are to be returned
• **include_empty** – Whether to return empty folders as well.
• **include_hidden** – Include folders that the user can’t see
• **preload_event** – in the process, preload all objects tied to the corresponding event and keep them in cache

```python
indico.modules.attachments.util.get_attached_items(linked_object, in-
clude_empty=True, in-
clude_hidden=True, pre-
load_event=False)
```

Return a structured representation of all the attachments linked to an object.

**Parameters**

• **linked_object** – The object whose attachments are to be returned
• **include_empty** – Whether to return empty folders as well.
• **include_hidden** – Include folders that the user can’t see
• **preload_event** – in the process, preload all objects tied to the corresponding event and keep them in cache

```python
indico.modules.attachments.util.get_default_folder_names()
```

```python
indico.modules.attachments.util.get_event(linked_object)
```

```python
indico.modules.attachments.util.get_nested_attached_items(obj)
```

Returns a structured representation of all attachments linked to an object and all its nested objects.

**Parameters**

`obj` – A Event, Session, Contribution or SubContribution object.

```python
class indico.modules.attachments.preview.ImagePreviewer
    Bases: indico.modules.attachments.preview.Previewer
    ALLOWED_CONTENT_TYPE = <_sre.SRE_Pattern object>
    TEMPLATE = u'image_preview.html'

class indico.modules.attachments.preview.MarkdownPreviewer
    Bases: indico.modules.attachments.preview.Previewer
    ALLOWED_CONTENT_TYPE = <_sre.SRE_Pattern object>
    classmethod generate_content(attachment)

class indico.modules.attachments.preview.PDFPreviewer
    Bases: indico.modules.attachments.preview.Previewer
    ALLOWED_CONTENT_TYPE = <_sre.SRE_Pattern object>
    TEMPLATE = u'iframe_preview.html'
    classmethod can_preview(attachment_file)

class indico.modules.attachments.preview.Previewer
    Bases: object
    Base class for file previewers
    To create a new file previewer, subclass this class and register it using the get_file_previewers signal.
    ALLOWED_CONTENT_TYPE = None
    TEMPLATE = None
```
classmethod can_preview(attachment_file)
    Checks if the content type of the file matches the allowed content type of files that the previewer can be used for.

classmethod generate_content(attachment)
    Generates the HTML output of the file preview

class indico.modules.attachments.preview.TextPreviewer
    Bases: indico.modules.attachments.preview.Previewer

    ALLOWED_CONTENT_TYPE = _sre.SRE_Pattern object

    classmethod generate_content(attachment)

    indico.modules.attachments.preview.get_file_previewer(attachment_file)
    Returns a file previewer for the given attachment file based on the file’s content type.

indico.modules.attachments.preview.get_file_previewers()

5.1.24 Room booking

Todo: Docstrings (module, models, utilities, services)

Models

class indico.modules.rb.models.rooms.Room(**kwars)
    Bases: indico.core.db.sqlalchemy.protection.ProtectionManagersMixin, sqlalchecy.ext.declarative.api.Model, indico.util.serializer.Serializer

    A simple constructor that allows initialization from kwars.

    Sets attributes on the constructed instance using the names and values in kwars.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    access_key = None

    acl_entries

    attributes

    available_equipment

    blocked_rooms

    bookable_hours

    booking_limit_days

    building

    can_access(user, allow_admin=True)

    can_book(user, allow_admin=True)

    can_delete(user)

    can_edit(user)
can_manage (user, permission=None, allow_admin=True, check_parent=True, explicit_permission=False)
can_moderate (user, allow_admin=True)
can_override (user, allow_admin=True)
can_prebook (user, allow_admin=True)
capacity
check_advance_days (end_date, user=None, quiet=False)
check_bookable_hours (start_time, end_time, user=None, quiet=False)
comments
default_protection_mode = 0
details_url
disallowed_protection_modes = frozenset([<ProtectionMode.inheriting: 1>])
division
daily_notification
monthly_notification
weekly_notification
end_notifications_enabled
favorite_of
static filter_available (start_dt, end_dt, repetition, include_blockings=True, include_pre_bookings=True, include_pending_blockings=False)
    Returns a SQLAlchemy filter criterion ensuring that the room is available during the given time.
static filter_bookable_hours (start_time, end_time)
static filter_nonbookable_periods (start_dt, end_dt)
classmethod find_all (*args, **kwargs)
    Retrieves rooms, sorted by location and full name
classmethod find_with_attribute (attribute)
    Search rooms which have a specific attribute
floor
full_name
generate_name ()
get_attribute_by_name (attribute_name)
get_attribute_value (name, default=None)
get_blocked_rooms (*dates, **kwargs)
classmethod get_permissions_for_user (user, allow_admin=True)
    Get the permissions for all rooms for a user.
    In case of multipass-based groups it will try to get a list of all groups the user is in, and if that’s not possible
    check the permissions one by one for each room (which may result in many group membership lookups).
    It is recommended to not call this in any place where performance matters and to memoize the result.
static get_with_data (*args, **kwargs)
has_attribute(attribute_name)
has_equipment(*names)
has_photo
id
is_auto_confirm
is_deleted
is_reservable
static is_user_admin(user)
key_location
latitude
location
location_id
location_name
longitude
map_url
max_advance_days
name
nonbookable_periods
notification_before_days
notification_before_days_monthly
notification_before_days_weekly
notification_emails
notifications_enabled
number
own_no_access_contact = None
owner
    The owner of the room. This is purely informational and does not grant any permissions on the room.
owner_id
photo
photo_id
protection_mode
protection_parent
reservations
reservations_need_confirmation
set_attribute_value(name, value)
site
sprite_position
surface_area
television
verbose_name
   Verbese name for the room (long)

```python
class indico.modules.rb.models.room_attributes.RoomAttribute(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    id
    is_hidden
    name
    room_associations
    title
```

```python
class indico.modules.rb.models.room_attributes.RoomAttributeAssociation(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    attribute
    attribute_id
    room
    room_id
    value
```

```python
class indico.modules.rb.models.room_bookable_hours.BookableHours(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    end_time
    fits_period(st, et)
    room
    room_id
    start_time
```
class indico.modules.rb.models.room_nonbookable_periods.NonBookablePeriod(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

end_dt
overlaps (st, et)

room
room_id
start_dt

class indico.modules.rb.models.blockings.Blocking(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

allowed
    A descriptor that presents a read/write view of an object attribute.

blocked_rooms

can_delete (user, allow_admin=True)
can_edit (user, allow_admin=True)
can_override (user, room=None, explicit_only=False, allow_admin=True)
    Check if a user can override the blocking
    The following persons are authorized to override a blocking: - the creator of the blocking - anyone on the blocking’s ACL - unless explicit_only is set: rb admins and room managers (if a room is given)

created_by_id
created_by_user
    The user who created this blocking.

created_dt
date
external_details_url
id
is_active_at (d)
reason
start_date

class indico.modules.rb.models.blocked_rooms.BlockedRoom(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
State
    alias of BlockedRoomState

approve (notify_blocker=True)
    Approve the room blocking, rejecting all colliding reservations/occurrences.

blocking_id
id

reject (user=None, reason=None)
    Reject the room blocking.

rejected_by
rejection_reason
room
room_id
state
state_name
```

```python
class indico.modules.rb.models.blocked_rooms.BlockedRoomState
    Bases: indico.util.struct.enum.RichIntEnum

    accepted = 1
    pending = 0
    rejected = 2
```

```python
class indico.modules.rb.models.blocking_principals.BlockingPrincipal(**kwargs)
    Bases: indico.core.db.sqlalchemy.principals.PrincipalMixin, sqlalchemy.ext.declarative.api.Model
```

A simple constructor that allows initialization from `kwargs`.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
blocking_id
category_role = None
category_role_id = None
email = None
event_role = None
event_role_id = None
id
ip_network_group = None
ip_network_group_id = None
local_group
```
local_group_id
multipass_group_name
multipass_group_provider
principal_backref_name = u'in_blocking_acls'
type
unique_columns = (u'blocking_id',)
user
user_id

class indico.modules.rb.models.equipment.EquipmentType(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    features
    id
    name
    rooms

class indico.modules.rb.models.locations.Location(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    id
    is_deleted
    map_url_template
    name
    room_name_format
        Translate Postgres’ format syntax (e.g. %I$s/%2$s-%3$s) to Python’s.
    rooms

class indico.modules.rb.models.map_areas.MapArea(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    bottom_right_latitude
bottom_right_longitude
id
is_default
name
top_left_latitude
top_left_longitude
class indico.modules.rb.models.photos.Photo(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
data
id

exception indico.modules.rb.models.reservations.ConflictingOccurrences
Bases: exceptions.Exception

class indico.modules.rb.models.reservations.RepeatFrequency
Bases: int, indico.util.struct.enum.IndicoEnum

DAY = 1
MONTH = 3
NEVER = 0
WEEK = 2

class indico.modules.rb.models.reservations.RepeatMapping
Bases: object

classmethod get_message(repeat_frequency, repeat_interval)
classmethod get_short_name(repeat_frequency, repeat_interval)

mapping = {
    (<RepeatFrequency.NEVER: 0>, 0): (u'Single reservation', None, u'none'),
    (<RepeatFrequency.DAY: 1>, 1): (u'Repeat every day', None, u'once a day'),
    (<RepeatFrequency.DAY: 1>, 2): (u'Repeat once every three days', None, u'every three days'),
    (<RepeatFrequency.DAY: 1>, 3): (u'Repeat once every four days', None, u'every four days'),
    (<RepeatFrequency.DAY: 1>, 4): (u'Repeat once every 5 days', None, u'every 5 days'),
    (<RepeatFrequency.DAY: 1>, 5): (u'Repeat once every 6 days', None, u'every 6 days'),
    (<RepeatFrequency.DAY: 1>, 6): (u'Repeat once every 7 days', None, u'weekly'),
    (<RepeatFrequency.WEEK: 2>, 1): (u'Repeat every week', None, u'weekly'),
    (<RepeatFrequency.WEEK: 2>, 2): (u'Repeat every two weeks', None, u'every two weeks'),
    (<RepeatFrequency.WEEK: 2>, 3): (u'Repeat every three weeks', None, u'every three weeks'),
    (<RepeatFrequency.MONTH: 3>, 1): (u'Repeat every month', None, u'monthly'),
    (<RepeatFrequency.MONTH: 3>, 2): (u'Repeat every two months', None, u'every two months'),
    (<RepeatFrequency.MONTH: 3>, 3): (u'Repeat every three months', None, u'every three months'),
    (<RepeatFrequency.MONTH: 3>, 4): (u'Repeat every four months', None, u'every four months'),
    (<RepeatFrequency.MONTH: 3>, 5): (u'Repeat every five months', None, u'every five months'),
    (<RepeatFrequency.MONTH: 3>, 6): (u'Repeat every six months', None, u'every six months'),
    (<RepeatFrequency.MONTH: 3>, 7): (u'Repeat every seven months', None, u'every seven months'),
    (<RepeatFrequency.MONTH: 3>, 8): (u'Repeat every eight months', None, u'every eight months'),
    (<RepeatFrequency.MONTH: 3>, 9): (u'Repeat every nine months', None, u'every nine months'),
    (<RepeatFrequency.MONTH: 3>, 10): (u'Repeat every 10 months', None, u'every 10 months'),
    (<RepeatFrequency.MONTH: 3>, 11): (u'Repeat every 11 months', None, u'every 11 months'),
    (<RepeatFrequency.MONTH: 3>, 12): (u'Repeat every 12 months', None, u'annually'),

accept(user, reason=None)
add_edit_log(edit_log)
booked_for_id
booked_for_name
booked_for_user

The user this booking was made for. Assigning a user here also updates booked_for_name.

booking_reason

can_accept (user, allow_admin=True)
can_cancel (user, allow_admin=True)
can_delete (user, allow_admin=True)
can_edit (user, allow_admin=True)
can_reject (user, allow_admin=True)
cancel (user, reason=None, silent=False)

contact_email

classmethod create_from_data (room, data, user, prebook=None, ignore_admin=False)

Creates a new reservation.

Parameters

• room – The Room that’s being booked.
• data – A dict containing the booking data, usually from a NewBookingConfirmForm instance
• user – The User who creates the booking.
• prebook – Instead of determining the booking type from the user’s permissions, always use the given mode.

create_occurrences (skip_conflicts, user=None)

created_by_id

created_by_user

The user who created this booking.

created_dt

edit_logs

der

dend_notification_sent

event

dexternal_details_url

dfind_excluded_days()

dfind_overlapping()

dstatic find_overlapping_with (room, occurrences, skip_reservation_id=None)

dget_conflicting_occurrences()

dstatic get_with_data (*args, **kwargs)

did

dis_accepted

dis_archived

dis_booked_for (user)
is_cancelled
is_owned_by(user)
is_pending
is_rejected
is_repeating
link
link_id
linked_object
location_name
modify(data, user)
    Modifies an existing reservation.

    Parameters
    
    • data – A dict containing the booking data, usually from a ModifyBookingForm instance
    
    • user – The User who modifies the booking.

occurrences
reject(user, reason, silent=False)
rejection_reason
repeat_frequency
repeat_interval
repetition
reset_approval(user)
room_id
start_dt
state

class indico.modules.rb.models.reservations.ReservationLink(**kwargs)
    Bases: indico.core.db.sqlalchemy.links.LinkMixin, sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in kwargs.

    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    allowed_link_types = set({<LinkType.event: 2>, <LinkType.contribution: 3>, <LinkType.session_block: 6>})
category = None
category_id = None
contribution
contribution_id
event
```

event_id

events_backref_name = u'all_room_reservation_links'

id

link_backref_name = u'room_reservation_links'

link_type

linked_event

linked_event_id

reservation

session = None

session_block

session_block_id

session_id = None

subcontribution = None

subcontribution_id = None

class indico.modules.rb.models.reservations.ReservationState

Bases: int, indico.util.struct.enum.IndicoEnum

accepted = 2

cancelled = 3

pending = 1

rejected = 4

class indico.modules.rb.models.reservation_edit_logs.ReservationEditLog(**kwargs)

Bases: sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

id

info

reservation

reservation_id

timestamp

user_name

class indico.modules.rb.models.reservation_occurrences.ReservationOccurrence(**kwargs)

Bases: sqlalchemy.ext.declarative.api.Model, indico.util.serializer.Serializer

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.
```
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
NO_RESERVATION_USER_STRATEGY = <sqlalchemy.orm.strategy_options._UnboundLoad object>
```

A relationship loading strategy that will avoid loading the users linked to a reservation. You want to use this in pretty much all cases where you eager-load the `reservation` relationship.

```python
can_cancel(user, allow_admin=True)
can_reject(user, allow_admin=True)
cancel(*args, **kwargs)
classmethod create_series(start, end, repetition)
classmethod create_series_for_reservation(reservation)
date
end_dt
external_cancellation_url
static filter_overlap(occurrences)
classmethod find_overlapping_with(room, occurrences, skip_reservation_id=None)
get_overlap(occurrence, skip_self=False)
is_cancelled
is_rejected
is_valid
is_within_cancel_grace_period
classmethod iter_create_occurrences(start, end, repetition)
static iter_start_time(start, end, repetition)
notification_sent
overlaps(occurrence, skip_self=False)
reject(*args, **kwargs)
rejection_reason
reservation
reservation_id
start_dt
state
class indico.modules.rb.models.reservation_occurrences.ReservationOccurrenceState
    Bases: int, indico.util.struct.enum.IndicoEnum
    cancelled = 3
    rejected = 4
    valid = 2
```

```python
indico.modules.rb.models.util.proxy_to_reservation_if_last_valid_occurrence(f)
```

Forwards a method call to `self.reservation` if there is only one occurrence.

5.1. API reference
Utilities

indico.modules.rb.util.TempReservationConcurrentOccurrence
    alias of indico.modules.rb.util.ReservationOccurrenceTmp

indico.modules.rb.util.TempReservationOccurrence
    alias of indico.modules.rb.util.ReservationOccurrenceTmp

indico.modules.rb.util.build_rooms_spritesheet()

indico.modules.rb.util.generate_spreadsheet_from_occurrences(occurrences)
    Generate spreadsheet data from a given booking occurrence list.
    Parameters
        occurrences – The booking occurrences to include in the spreadsheet

indico.modules.rb.util.get_booking_params_for_event(event)
    Get a set of RB interface parameters suitable for this event.
    These parameters can then be used to construct a URL that will lead to a pre-filled search that matches the
    start/end times for a given day.
    Parameters
        event – Event object

indico.modules.rb.util.get_linked_object(type_, id_)

indico.modules.rb.util.get_resized_room_photo(room)

indico.modules.rb.util.group_by_occurrence_date(occurrences, sort_by=None)

indico.modules.rb.util.is_booking_start_within_grace_period(start_dt, user, allow_admin=False)

indico.modules.rb.util.rb_check_user_access(*args, **kwargs)
    Checks if the user has access to the room booking system

indico.modules.rb.util.rb_is_admin(*args, **kwargs)
    Checks if the user is a room booking admin

indico.modules.rb.util.remove_room_spritesheet_photo(room)

indico.modules.rb.util.serialize_availability(availability)

indico.modules.rb.util.serialize_blockings(data)

indico.modules.rb.util.serialize_booking_details(booking)

indico.modules.rb.util.serialize_concurrent_pre_bookings(data)

indico.modules.rb.util.serialize_nonbookable_periods(data)

indico.modules.rb.util.serialize_occurrences(data)

indico.modules.rb.util.serialize_unbookable_hours(data)

indico.modules.rb.statistics.calculate_rooms_bookable_time(rooms,
    start_date=None, end_date=None)

indico.modules.rb.statistics.calculate_rooms_booked_time(rooms, start_date=None, end_date=None)

indico.modules.rb.statistics.calculate_rooms_occupancy(rooms, start=None, end=None)
5.1.25 Authentication

Todo: Docstrings (module, models, utilities)

Models

class indico.modules.auth.models.identities.Identity(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

Identities of Indico users

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

    data
    id
        the unique id of the identity

    identifier
        the unique identifier of the user within its provider

    last_login_dt
        the timestamp of the latest login

    last_login_ip
        the ip address that was used for the latest login

    locator

    multipass_data
        internal data used by the flask-multipass system

    password
        the password of the user in case of a local identity

    password_hash
        the hash of the password in case of a local identity

    provider
        the provider name of the identity

    register_login(ip)
        Updates the last login information

    safe_last_login_dt
        last_login_dt that is safe for sorting (no None values)

    user_id
        the id of the user this identity belongs to

class indico.modules.auth.models.registration_requests.RegistrationRequest(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

```python
comment
email
extra_emails
id
identity_data
locator
```

Defines a smart locator property.

This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.

This decorator should usually be applied to a method named `locator` as this name is required for `get_locator` to find it automatically when just passing the object.

If you need more than one locator, you can define it like this:

```python
@locator_property
def locator(self):
    return {...}

@locator
def locator(self):
    return {...}
```

The `other` locator can then be accessed by passing `obj.locator.other` to the code expecting an object with a locator.

```python
settings
user_data
```

Utilities

```python
indico.modules.auth.util.impersonate_user(user)
Impersonate another user as an admin

indico.modules.auth.util.load_identity_info()
Retrieves identity information from the session

indico.modules.auth.util.redirect_to_login(next_url=None, reason=None)
Redirects to the login page.

Parameters

- **next_url** – URL to be redirected upon successful login. If not specified, it will be set to `request.relative_url`.
- **reason** – Why the user is redirected to a login page.

indico.modules.auth.util.register_user(email, extra_emails, user_data, identity_data, settings, from_moderation=False)
Create a user based on the registration data provided during the user registration process (via `RHHRegister` and `RegistrationHandler`).

272 Chapter 5. API reference
This method is not meant to be used for generic user creation, the only reason why this is here is that approving a registration request is handled by the `users` module.

```python
indico.modules.auth.util.save_identity_info(identity_info, user)
```
Saves information from IdentityInfo in the session

```python
indico.modules.auth.util.undo_impersonate_user()
```
Undo an admin impersonation login and revert to the old user

```python
indico.modules.auth.util.url_for_login(next_url=None)
indico.modules.auth.util.url_for_logout(next_url=None)
indico.modules.auth.util.url_for_register(next_url=None, email=None)
```
Returns the URL to register

**Parameters**

- **next_url** – The URL to redirect to afterwards.
- **email** – A pre-validated email address to use when creating a new local account. Use this argument ONLY when sending the link in an email or if the email address has already been validated using some other way.

### 5.1.26 OAuth

*Todo:* Docstrings (module, models, provider)

#### Models

```python
class indico.modules.oauth.models.applications.OAuthApplication(**kwargs)
```
Bases: `sqlalchemy.ext.declarative.api.Model`

OAuth applications registered in Indico

A simple constructor that allows initialization from `kwargs`.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

- **client_id**
  - the OAuth client_id

- **client_secret**
  - the OAuth client_secret

- **client_type**
- **default_redirect_uri**
- **default_scopes**
  - the OAuth default scopes the application may request access to

- **description**
  - human readable description

- **id**
  - the unique id of the application
is_enabled
whether the application is enabled or disabled

is_trusted
whether the application can access user data without asking for permission

locator

name
human readable name

redirect_uris
the OAuth absolute URIs that a application may use to redirect to after authorization

reset_client_secret()

system_app_type
the type of system app (if any). system apps cannot be deleted

validate_redirect_uri(redirect_uri)
Called by flask-oauthlib to validate the redirect_uri.

Uses a logic similar to the one at GitHub, i.e. protocol and host/port must match exactly and if there is a path in the whitelisted URL, the path of the redirect_uri must start with that path.

class indico.modules.oauth.models.applications.SystemAppType
Bases: int, indico.util.struct.enum.IndicoEnum

checkin = 1

default_data

enforced_data

flower = 2

none = 0

class indico.modules.oauth.models.tokens.OAuthGrant(client_id, code, redirect_uri, user, scopes, expires)
Bases: object

OAuth grant token

delete()

classmethod get(client_id, code)

key

classmethod make_key(client_id, code)

save()

ttl

class indico.modules.oauth.models.tokens.OAuthToken(**kwars)
Bases: sqlalchemy.ext.declarative.api.Model

OAuth tokens

A simple constructor that allows initialization from kwars.

Sets attributes on the constructed instance using the names and values in kwars.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
access_token
   an unguessable unique string of characters

application
   application authorized by this token

application_id
   the identifier of the linked application

expires

id
   the unique identifier of the token

last_used_dt
   the last time the token was used by the application

locator

scopes
   The set of scopes the linked application has access to.

type

user
   the user who owns this token

user_id
   the identifier of the linked user

Utilities

exception indico.modules.oauth.provider.DisabledClientIdError (description=None, uri=None, state=None, status_code=None, request=None)

Bases: oauthlib.oauth2.rfc6749.errors.FatalClientError

description: A human-readable ASCII [USASCII] text providing additional information, used to assist the client developer in understanding the error that occurred. Values for the “error_description” parameter MUST NOT include characters outside the set x20-21 / x23-5B / x5D-7E.

uri: A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. Values for the “error_uri” parameter MUST conform to the URI-Reference syntax, and thus MUST NOT include characters outside the set x21 / x23-5B / x5D-7E.

state: A CSRF protection value received from the client.

request: Oauthlib Request object

error = u'application_disabled_by_admin'

indico.modules.oauth.provider.load_client (client_id)
indico.modules.oauth.provider.load_grant (client_id, code)
indico.modules.oauth.provider.load_token (access_token, refresh_token=None)
indico.modules.oauth.provider.save_grant (client_id, code, request, *args, **kwargs)
indico.modules.oauth.provider.save_token (token_data, request, *args, **kwargs)
5.1.27 Group

**Todo:** Docstrings (module)

### Models

**class** `indico.modules.groups.models.groups.LocalGroup(**kwargs)`

**Bases:** `sqlalchemy.ext.declarative.api.Model`

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

- `id`:
  the unique id of the group

- `in_attachment_folder_acls`
- `in_room_acls`
- `members`:
  the users in the group

- `name`:
  the name of the group

- `proxy`:
  Returns a GroupProxy wrapping this group

**class** `indico.modules.groups.core.GroupProxy`

**Bases:** `object`

Provides a generic interface for both local and multipass groups.

Creating an instance of this class actually creates either a `LocalGroupProxy` or a `MultipassGroupProxy`, but they expose the same API.

**Parameters**

- `name_or_id` – The name of a multipass group or ID of a local group
- `provider` – The provider of a multipass group

Creates the correct GroupProxy for the group type

- `as_legacy`

- `as_legacy_group`:
  The legacy-style group wrapper

- `as_principal`:
  The serializable principal identifier of this group

- `get_members()`:
  Gets the list of users who are members of the group
classmethod get_named_default_group(name)
    Gets the group with the matching name from the default group provider.

    If there is no default group provider, local groups will be used and name is the group’s ID.

    This method should only be used for legacy code or code that gets the group name from an external source
    which does not contain a provider identifier.

    group
    The underlying group object

has_member(user)
    Checks if the user is a member of the group.

    This can also be accessed using the in operator.

identifier
    is_category_role = False
    is_event_role = False
    is_group = True
    is_network = False
    is_single_person = False
    principal_order = 3

classmethod search(name, exact=False, providers=None)
    Searches for groups

    Parameters
    • name – The group name to search for.
    • exact – If only exact matches should be found (much faster)
    • providers – None to search in all providers and local groups. May be a set specifying
      providers to search in. For local groups, the 'indico' provider name may be used.

Utilities

indico.modules.groups.util.serialize_group(group)
    Serialize group to JSON-like object

5.1.28 Video conference

Todo: Docstrings (module, models, utilities, plugins, exceptions)

Models

class indico.modules.vc.models.vc_rooms.VCRoom(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    A simple constructor that allows initialization from kwargs.

    Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.

`created_by_id`
  ID of the creator

`created_by_user`
  The user who created the videoconference room

`created_dt`
  Creation timestamp of the videoconference room

`data`
  videoconference plugin-specific data

`id`
  Videoconference room ID

`locator`

`modified_dt`
  Modification timestamp of the videoconference room

`name`
  Name of the videoconference room

`plugin`

`status`
  Status of the videoconference room

`type`
  Type of the videoconference room

```python
class indico.modules.vc.models.vc_rooms.VCRoomEventAssociation(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Model
```

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any
mapped columns or relationships.

`contribution_id`

`data`
  videoconference plugin-specific data

`delete(user, delete_all=False)`
  Deletes a VC room from an event

  If the room is not used anywhere else, the room itself is also deleted.

  **Parameters**

  - `user` – the user performing the deletion
  - `delete_all` – if True, the room is detached from all events and deleted.

`event`
  The associated Event

`event_id`
  ID of the event
classmethod find_for_event(event, include_hidden=False, include_deleted=False, only_linked_to_event=False, **kwargs)

Returns a Query that retrieves the videoconference rooms for an event.

Parameters

- **event** – an indico Event
- **only_linked_to_event** – only retrieve the vc rooms linked to the whole event
- **kwargs** – extra kwargs to pass to find()

classmethod get_linked_for_event(**kwargs)

Get a dict mapping link objects to event vc rooms

id

Association ID

link_object

link_type

Type of the object the vc_room is linked to

linked_block

The linked session block (if the VC room is attached to a block)

linked_contrib

The linked contribution (if the VC room is attached to a contribution)

linked_event

The linked event (if the VC room is attached to the event itself)

linked_event_id

locator

classmethod register_link_events()

session_block_id

show

If the vc room should be shown on the event page

vc_room

The associated :class:VCRoom

vc_room_id

ID of the videoconference room

class indico.modules.vc.models.vc_rooms.VCRoomLinkType

Bases: int, indico.util.struct.enum.IndicoEnum

block = 3

collection = 2

event = 1

class indico.modules.vc.models.vc_rooms.VCRoomStatus

Bases: int, indico.util.struct.enum.IndicoEnum

created = 1

deleted = 2
Utilities

indico.modules.vc.util.find_event_vc_rooms(from_dt=None, to_dt=None, distinct=False)
Finds VC rooms matching certain criteria

Parameters
- **from_dt** – earliest event/contribution to include
- **to_dt** – latest event/contribution to include
- **distinct** – if True, never return the same (event, vcroom) more than once (even if it’s linked more than once to that event)

indico.modules.vc.util.get_linked_to_description(obj)

indico.modules.vc.util.get_managed_vc_plugins(user)
Returns the plugins the user can manage

indico.modules.vc.util.get_vc_plugins()
Returns a dict containing the available videoconference plugins.

indico.modules.vc.util.resolve_title(obj)

Plugins

class indico.modules.vc.plugins.VCPluginMixin
Bases: object

    acl_settings = set([u'acl', u'managers'])

can_manage_vc(user)
Checks if a user has management rights on this VC system

can_manage_vc_room(user, room)
Checks if a user can manage a vc room

can_manage_vc_rooms(user, event)
Checks if a user can manage vc rooms on an event

category = u'Videoconference'

create_form(event, existing_vc_room=None, existing_event_vc_room=None)
Creates the videoconference room form

Parameters
- **event** – the event the videoconference room is for
- **existing_vc_room** – a vc_room from which to retrieve data for the form

Returns an instance of an IndicoForm subclass

create_room(vc_room, event)

default_settings = {u'notification_emails': []}

friendly_name = None
the readable name of the VC plugin

get_notification_bcc_list(action, vc_room, event)
get_notification_cc_list(action, vc_room, event)
get_vc_room_attach_form_defaults(event)
get_vc_room_form_defaults(event)
icon_url
init()
logo_url

render_buttons(vc_room, event_vc_room, **kwargs)
Renders a list of plugin specific buttons (eg: Join URL, etc) in the management area

Parameters
• vc_room – the VC room object
• event_vc_room – the association of an event and a VC room
• kwargs – arguments passed to the template

render_event_buttons(vc_room, event_vc_room, **kwargs)
Renders a list of plugin specific buttons (eg: Join URL, etc) in the event page

Parameters
• vc_room – the VC room object
• event_vc_room – the association of an event and a VC room
• kwargs – arguments passed to the template

render_form(**kwargs)
Renders the videoconference room form :param kwargs: arguments passed to the template

render_info_box(vc_room, event_vc_room, event, **kwargs)
Renders the information shown in the expandable box of a VC room row :param vc_room: the VC room object :param event_vc_room: the association of an event and a VC room :param event: the event with the current VC room attached to it :param kwargs: arguments passed to the template

render_manage_event_info_box(vc_room, event_vc_room, event, **kwargs)
Renders the information shown in the expandable box on a VC room in the management area

Parameters
• vc_room – the VC room object
• event_vc_room – the association of an event and a VC room
• event – the event with the current VC room attached to it
• kwargs – arguments passed to the template

service_name
settings_form
alias of indico.modules.vc.forms.VCPluginSettingsFormBase

update_data_association(event, vc_room, event_vc_room, data)
update_data_vc_room(vc_room, data)

vc_room_attach_form = None
the IndicoForm to use for the videoconference room attach form

vc_room_form = None
the IndicoForm to use for the videoconference room form
Exceptions

exception indico.modules.vc.exceptions.VCRoomError(message, field=None)  
Bases: exceptions.Exception

exception indico.modules.vc.exceptions.VCRoomNotFoundError(message)  
Bases: indico.modules.vc.exceptions.VCRoomError

5.1.29 Designer

Todo: Docstrings (module, models, utilities)

Models

class indico.modules.designer.models.images.DesignerImageFile(**kwargs)  
Bases: indico.core.storage.models.StoredFileMixin, sqlalchemy.ext.declarative.api.Model

A simple constructor that allows initialization from kwargs.
Sets attributes on the constructed instance using the names and values in kwargs.
Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.

content_type  
The MIME type of the file

created_dt  
The date/time when the file was uploaded

download_url

filename  
The name of the file

id  
The ID of the file

locator

md5  
An MD5 hash of the file.
Automatically assigned when save() is called.

size  
The size of the file (in bytes).
Automatically assigned when save() is called.

storage_backend

storage_file_id

template

template_id  
The designer template the image belongs to
version_of = None

class indico.modules.designer.models.templates.DesignerTemplate(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model

    background_image
    background_image_id
    backside_template
    backside_template_id
    category
    category_id
    data
    default_ticket_template_of
    event
    event_id
    id
    is_clonable
    is_system_template
    is_ticket

    locator
        Defines a smart locator property.
        This behaves pretty much like a normal read-only property and the decorated function should return a dict
        containing the necessary data to build a URL for the object.
        This decorator should usually be applied to a method named locator as this name is required for
        get_locator to find it automatically when just passing the object.
        If you need more than one locator, you can define it like this:

        ```python
        @locator_property
        def locator(self):
            return {...}
        @locator
        def locator(self):
            return {...}
        ```
        The other locator can then be accessed by passing obj.locator.other to the code expecting an
        object with a locator.

    owner
    title
    type

Utilities

indico.modules.designer.util.get_all_templates(obj)
    Get all templates usable by an event/category
indico.modules.designer.util.get_default_template_on_category(category,
    only_inherited=False)

indico.modules.designer.util.get_inherited_templates(obj)
    Get all templates inherited by a given event/category

indico.modules.designer.util.get_nested_placeholder_options()

indico.modules.designer.util.get_not_deletable_templates(obj)
    Get all non-deletable templates for an event/category

indico.modules.designer.util.get_placeholder_options()

class indico.modules.designer.pdf.DesignerPDFBase(template, config)
    Bases: object
    get_pdf()

class indico.modules.designer.pdf.TplData(width, height, items, background_position,
    width_cm, height_cm)
    Bases: tuple
    Create new instance of TplData(width, height, items, background_position, width_cm, height_cm)
    background_position
        Alias for field number 3
    height
        Alias for field number 1
    height_cm
        Alias for field number 5
    items
        Alias for field number 2
    width
        Alias for field number 0
    width_cm
        Alias for field number 4

Placeholders

class indico.modules.designer.placeholders.EventDatesPlaceholder
    Bases: indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Event Dates'
    group = u'event'
    name = u'event_dates'
    classmethod render(event)

class indico.modules.designer.placeholders.EventDescriptionPlaceholder
    Bases: indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Event Description'
    group = u'event'
    name = u'event_description'
    classmethod render(event)
class indico.modules.designer.placeholders.RegistrationFullNamePlaceholder
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase

    description = lu'Full Name'
    name = u'full_name'
    name_options = {}
    with_title = True

class indico.modules.designer.placeholders.EventOrgTextPlaceholder
    Bases: indico.modules.designer.placeholders.DesignerPlaceholder

    description = lu'Event Organizers'
    group = u'event'
    name = u'event_organizers'
    classmethod render(event)

class indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholder
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase

    description = lu'Full Name (no title)'
    name = u'full_name_no_title'
    name_options = {}
    with_title = False

class indico.modules.designer.placeholders.RegistrationFullNamePlaceholderB
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase

    description = lu'Full Name B'
    name = u'full_name_b'
    name_options = {u'last_name_first': False}
    with_title = True

class indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderB
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase

    description = lu'Full Name B (no title)'
    name = u'full_name_b_no_title'
    name_options = {u'last_name_first': False}
    with_title = False

class indico.modules.designer.placeholders.RegistrationFullNamePlaceholderC
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase

    description = lu'Full Name C'
    name = u'full_name_c'
    name_options = {u'last_name_first': False, u'last_name_upper': True}
    with_title = True

class indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderC
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase

5.1. API reference
description = lu'Full Name C (no title)'
name = u'full_name_no_title_c'
name_options = {u'last_name_upper': True}
with_title = False

class indico.modules.designer.placeholders.RegistrationFullNamePlaceholderD
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase
    description = lu'Full Name D (abbrev.)'
    name = u'full_name_d'
    name_options = {u'abbrev_first_name': True, u'last_name_first': False, u'last_name_upper': True}
    with_title = True

class indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderD
    Bases: indico.modules.designer.placeholders.FullNamePlaceholderBase
    description = lu'Full Name D (abbrev., no title)'
    name = u'full_name_no_title_d'
    name_options = {u'abbrev_first_name': True, u'last_name_upper': True}
    with_title = False

class indico.modules.designer.placeholders.RegistrationTitlePlaceholder
    Bases: indico.modules.designer.placeholders.RegistrationPDPlaceholder
    description = lu'Title'
    field = u'title'
    name = u'title'

class indico.modules.designer.placeholders.RegistrationFirstNamePlaceholder
    Bases: indico.modules.designer.placeholders.RegistrationPlaceholder
    description = lu'First Name'
    field = u'first_name'
    name = u'first_name'

class indico.modules.designer.placeholders.RegistrationLastNamePlaceholder
    Bases: indico.modules.designer.placeholders.RegistrationPlaceholder
    description = lu'Last Name'
    field = u'last_name'
    name = u'last_name'

class indico.modules.designer.placeholders.RegistrationTicketQRPlaceholder
    Bases: indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Ticket QR Code'
    group = u'registrant'
    is_ticket = True
    name = u'ticket_qr_code'
    classmethod render(registration)
class Indico.Modules.Designer.Placeholders.RegistrationEmailPlaceholder
    description = lu'E-mail'
    field = u'email'
    name = u'email'

class Indico.Modules.Designer.Placeholders.RegistrationAmountPlaceholder
    description = lu'Price (no currency)'
    name = u'amount'
    classmethod render(registration)

class Indico.Modules.Designer.Placeholders.RegistrationPricePlaceholder
    description = lu'Price (with currency)'
    name = u'price'
    classmethod render(registration)

class Indico.Modules.Designer.Placeholders.RegistrationFriendlyIDPlaceholder
    description = lu'Registration ID'
    field = u'friendly_id'
    name = u'registration_friendly_id'

class Indico.Modules.Designer.Placeholders.RegistrationAffiliationPlaceholder
    description = lu'Institution'
    field = u'affiliation'
    name = u'affiliation'

    description = lu'Position'
    field = u'position'
    name = u'position'

class Indico.Modules.Designer.Placeholders.RegistrationAddressPlaceholder
    description = lu'Address'
    field = u'address'
    name = u'address'

    description = lu'Country'
    field = u'country'
class indico.modules.designer.placeholders.RegistrationPhonePlaceholder
    name = u'country'
    class
    indico.modules.designer.placeholders.RegistrationPDPlaceholder
    description = lu'Phone'
    field = u'phone'
    name = u'phone'

class indico.modules.designer.placeholders.EventTitlePlaceholder
    name = u'event_title'
    class
    indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Event Title'
    group = u'event'
    name = u'event_title'
    classmethod render(event)

class indico.modules.designer.placeholders.CategoryTitlePlaceholder
    name = u'category_title'
    class
    indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Category Title'
    group = u'event'
    name = u'category_title'
    classmethod render(event)

class indico.modules.designer.placeholders.EventRoomPlaceholder
    name = u'event_room'
    class
    indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Event Room'
    group = u'event'
    name = u'event_room'
    classmethod render(event)

class indico.modules.designer.placeholders.EventVenuePlaceholder
    name = u'event_venue'
    class
    indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Event Venue'
    group = u'event'
    name = u'event_venue'
    classmethod render(event)

class indico.modules.designer.placeholders.EventSpeakersPlaceholder
    name = u'event_speakers'
    class
    indico.modules.designer.placeholders.DesignerPlaceholder
    description = lu'Event Speakers/Chairs'
    group = u'event'
    name = u'event_speakers'
    classmethod render(event)
5.1.30 Network

Todo: Docstrings (module, models)

Models

class indico.modules.networks.models.networks.IPNetwork(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    group_id
    network

class indico.modules.networks.models.networks.IPNetworkGroup(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    attachment_access_override
        Grants all IPs in the network group read access to all attachments
    contains_ip(ip)
    description
    hidden
        Whether the network group is hidden in ACL forms
    id
    is_category_role = False
    is_event_role = False
    is_group = False
    is_network = True
    is_single_person = False
    locator
    name
    networks
        A descriptor that presents a read/write view of an object attribute.
        principal_order = 1
        principal_type = 5
Utilities

```python
indico.modules.networks.util.serialize_ip_network_group(group)
```
Serialize group to JSON-like object

5.1.31 News

Todo: Docstrings (module, models)

Models

```python
class indico.modules.news.models.news.NewsItem(**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Model
    A simple constructor that allows initialization from kwargs.
    Sets attributes on the constructed instance using the names and values in kwargs.
    Only keys that are present as attributes of the instance’s class are allowed. These could be, for example, any mapped columns or relationships.
    anchor
    content
    created_dt
    id
    locator
        Defines a smart locator property.
        This behaves pretty much like a normal read-only property and the decorated function should return a dict containing the necessary data to build a URL for the object.
        This decorator should usually be applied to a method named locator as this name is required for get_locator to find it automatically when just passing the object.
        If you need more than one locator, you can define it like this:
        ```
        @locator_property
        def locator(self):
            return {...}
        @locator.other
        def locator(self):
            return {...}
        ```
        The other locator can then be accessed by passing obj.locator.other to the code expecting an object with a locator.
    title
```
Utilities

```python
indico.modules.news.util.get_recent_news(*args, **kwargs)
```

Get a list of recent news for the home page

## 5.1.32 Indico fields

**Todo:** Docstrings to all fields

Indico fields extend from WTForm fields and are used for the special cases where the simple form fields are not enough to cover all needs.

```python
class indico.modules.events.fields.EventPersonLinkListField(*args, **kwargs)
    Bases: indico.modules.events.fields.PersonLinkListFieldBase
    A field to manage event’s chairpersons
    ```

```python
linked_object_attr = u'event'
```

```python
person_link_cls
    alias of indico.modules.events.models.persons.EventPersonLink
```

```python
pre_validate(form)
```

```python
widget = <indico.web.forms.widgets.JinjaWidget object>
```

```python
class indico.modules.events.fields.EventPersonListField(*args, **kwargs)
    Bases: indico.web.forms.fields.principals.PrincipalListField
    A field that lets you select a list Indico user and EventPersons
    Requires its form to have an event set.
    ```

```python
create_untrusted_persons = False
    Whether new event persons created by the field should be marked as untrusted
```

```python
event
pre_validate(form)
```

```python
process_formdata(valuelist)
```

```python
class indico.modules.events.fields.IndicoThemeSelectField(*args, **kwargs)
    Bases: wtforms.fields.core.SelectField
```

```python
class indico.modules.events.fields.PersonLinkListFieldBase(*args, **kwargs)
    Bases: indico.modules.events.fields.EventPersonListField
    default_sort_alpha = True
        If set to True, will be sorted alphabetically by default
    ```

```python
linked_object_attr = None
    name of the attribute on the form containing the linked object
```

```python
person_link_cls = None
    class that inherits from PersonLinkBase
```

```python
widget = None
```

```python
class indico.modules.events.fields.RatingReviewField(*args, **kwargs)
    Bases: wtforms.fields.core.RadioField
```

5.1. API reference
A field to manage external references.

```python
def pre_validate(form):
    pass
def process_formdata(valuelist):
    pass
```

```python
class ReferencesField(*args, **kwargs):
    Bases: MultipleItemsField
    A field to manage external references.
    def pre_validate(form):
        pass
    def process_formdata(valuelist):
        pass
```

### AbstractField

A selectize-based field to select an abstract from an event.

```python
def event
    return AbstractField(*args, **kwargs)
```

```python
class AbstractField(*args, **kwargs):
    Bases: QuerySelectField
    A selectize-based field to select an abstract from an event.
    def event
        return AbstractField(*args, **kwargs)
```

### AbstractPersonLinkListField

A field to configure a list of abstract persons

```python
def create_untrusted_persons = True
default_sort_alpha = False
linked_object_attr = u'abstract'
person_link_cls
    alias of AbstractPersonLink
```

```python
def pre_validate(form):
    pass
```

```python
class AbstractPersonLinkListField(*args, **kwargs):
    Bases: AbstractPersonLinkListFieldBase
    A field to configure a list of abstract persons
    def create_untrusted_persons = True
default_sort_alpha = False
linked_object_attr = u'abstract'
person_link_cls
    alias of AbstractPersonLink
    def pre_validate(form):
        pass
```

### EmailRuleListField

A field that stores a list of e-mail template rules.

```python
def label=None,
    validators=None,
    filters=(),
    description=u'",
    id=None,
    default=None,
    widget=None,
    render_kw=None,
    _form=None,
    _name=None,
    _prefix=u'",
    _translations=None,
    _meta=None)
```

```python
class EmailRuleListField(*label=None,
    validators=None,
    filters=(),
    description=u'",
    id=None,
    default=None,
    widget=None,
    render_kw=None,
    _form=None,
    _name=None,
    _prefix=u'",
    _translations=None,
    _meta=None):
    Bases: JSONField
    A field that stores a list of e-mail template rules.
    ```
Construct a new field.

Parameters

- **label** – The label of the field.
- **validators** – A sequence of validators to call when *validate* is called.
- **filters** – A sequence of filters which are run on input data by *process*.
- **description** – A description for the field, typically used for help text.
- **id** – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.
- **default** – The default value to assign to the field, if no form or object input is provided. May be a callable.
- **widget** – If provided, overrides the widget used to render the field.
- **render_kw** (dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.
- **_form** – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.
- **_name** – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.
- **_prefix** – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.
- **_translations** – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.
- **_meta** – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If **_form** and **_name** isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

**CAN_POPULATE** = True

**accepted_condition_types** = (class 'indico.modules.events.abstracts.notifications.StateCondition',
    class 'indico.modules.events.abstracts.notifications.ContributionTypeCondition',
    class 'indico.modules.events.abstracts.notifications.TrackCondition',
    )

**condition_class_map** = {u'contribution_type': class 'indico.modules.events.abstracts.notifications.ContributionTypeCondition',
    u'track': class 'indico.modules.events.abstracts.notifications.TrackCondition',
    }

**pre_validate** (form)

**widget** = <indico.web.forms.widgets.JinjaWidget object>
class indico.modules.events.abstracts.fields.TrackRoleField(label=None, validators=None, filters=(), description=u"", id=None, default=None, widget=None, render_kw=None, _form=None, _name=None, _prefix=u"", _translations=None, _meta=None)

Bases: indico.web.forms.fields.simple.JSONField

A field that stores a list of e-mail template rules.

Construct a new field.

Parameters

- label – The label of the field.
- validators – A sequence of validators to call when validate is called.
- filters – A sequence of filters which are run on input data by process.
- description – A description for the field, typically used for help text.
- id – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.
- default – The default value to assign to the field, if no form or object input is provided. May be a callable.
- widget – If provided, overrides the widget used to render the field.
- render_kw (dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.
- _form – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.
- _name – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.
- _prefix – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.
- _translations – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.
- _meta – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If _form and _name isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

CAN_POPULATE = True
category_roles
event_roles
permissions_info
widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.modules.events.contributions.fields.ContributionPersonLinkListField(*args, **kwargs)

Bases: indico.modules.events.fields.PersonLinkListFieldBase

A field to configure a list of contribution persons

linked_object_attr = u'contrib'

person_link_cls
    alias of indico.modules.events.contributions.models.persons.ContributionPersonLink

pre_validate(form)

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.modules.events.contributions.fields.SubContributionPersonLinkListField(*args, **kwargs)

Bases: indico.modules.events.contributions.fields.ContributionPersonLinkListField

A field to configure a list of subcontribution persons

linked_object_attr = u'subcontrib'

person_link_cls
    alias of indico.modules.events.contributions.models.persons.SubContributionPersonLink

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.modules.events.papers.fields.PaperEmailSettingsField(label=None, validators=None, filters=(), description=u'', id=None, default=None, widget=None, render_kw=None, _form=None, _name=None, _prefix=u'', _translations=None, _meta=None)

Bases: indico.web.forms.fields.simple.JSONField

Construct a new field.

Parameters

- label – The label of the field.
- validators – A sequence of validators to call when validate is called.
- filters – A sequence of filters which are run on input data by process.
- description – A description for the field, typically used for help text.

5.1. API reference 295
• id – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.

• default – The default value to assign to the field, if no form or object input is provided. May be a callable.

• widget – If provided, overrides the widget used to render the field.

• render_kw\( (dict) \) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.

• _form – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.

• _name – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.

• _prefix – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.

• _translations – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.

• _meta – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If _form and _name isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

CAN_POPULATE = True

event

process_formdata\( (valuelist) \)

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.modules.events.sessions.fields.SessionBlockPersonLinkListField(*args, **kwargs)

Bases: indico.modules.events.fields.PersonLinkListFieldBase

linked_object_attr = u'session_block'

person_link_cls

alias of indico.modules.events.sessions.models.persons.

SessionBlockPersonLink

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.modules.categories.fields.CategoryField(*args, **kwargs)

Bases: wtforms.fields.simple.HiddenField

WTForms field that lets you select a category.

Parameters

• allow_events – Whether to allow selecting a category that contains events.

• allow_subcats – Whether to allow selecting a category that contains subcategories.

• require_event_creation_rights – Whether to allow selecting only categories where the user can create events.

pre_validate\( (form) \)

process_data\( (value) \)
process_formdata(valuelist)

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.modules.networks.fields.MultiIPNetworkField(*args, **kwargs)
    Bases: indico.web.forms.fields.itemlists.MultiStringField

    A field to enter multiple IPv4 or IPv6 networks.

    The field data is a set of IPNetwork's not bound to a DB session. The 'unique' and
    'sortable' parameters of the parent class cannot be used with this class.

pre_validate(form)
process_data(value)
process_formdata(valuelist)

class indico.web.forms.fields.IndicoSelectMultipleCheckboxField(label=None, validators=None, coerce=<type 'unicode'>, choices=None, **kwargs)
    Bases: wtforms.fields.core.SelectMultipleField

    option_widget = <wtforms.widgets.core.CheckboxInput object>
widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.IndicoRadioField(*args, **kwargs)
    Bases: wtforms.fields.core.RadioField

    widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.JSONField(label=None, validators=None, filters=(), description=u'', id=None, default=None, widget=None, render_kw=None, _form=None, _name=None, _prefix=u'', _translations=None, _meta=None)
    Bases: wtforms.fields.simple.HiddenField

    Construct a new field.

    Parameters

    - label – The label of the field.
    - validators – A sequence of validators to call when validate is called.
    - filters – A sequence of filters which are run on input data by process.
    - description – A description for the field, typically used for help text.
    - id – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.
    - default – The default value to assign to the field, if no form or object input is provided. May be a callable.
    - widget – If provided, overrides the widget used to render the field.
    - render_kw(dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.
• **_form** – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.

• **_name** – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.

• **_prefix** – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.

• **_translations** – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.

• **_meta** – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If `_form` and `_name` isn’t provided, an UnboundField will be returned instead. Call its `bind()` method with a form instance and a name to construct the field.

**CAN_POPULATE = False**

Whether an object may be populated with the data from this field

```python
populate_obj(obj, name)
```

```python
process_formdata(valuelist)
```

**class indico.web.forms.fields.HiddenFieldList**(label=None, validators=None, filters=(), description=u'', id=None, default=None, widget=None, render_kw=None, _form=None, _name=None, _prefix=u'', _translations=None, _meta=None)

Bases: wtforms.fields.simple.HiddenField

A hidden field that handles lists of strings.

This is done `getlist`-style, i.e. by repeating the input element with the same name for each list item.

The only case where this field is useful is when you display a form via POST and provide a list of items (e.g. ids) related to the form which needs to be kept when the form is submitted and also need to access it via `request.form.getlist(...)` before submitting the form.

Construct a new field.

**Parameters**

• **label** – The label of the field.

• **validators** – A sequence of validators to call when `validate` is called.

• **filters** – A sequence of filters which are run on input data by `process`.

• **description** – A description for the field, typically used for help text.

• **id** – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.

• **default** – The default value to assign to the field, if no form or object input is provided. May be a callable.

• **widget** – If provided, overrides the widget used to render the field.

• **render_kw**(dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.

• **_form** – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.
• _name – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.

• _prefix – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.

• _translations – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.

• _meta – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If _form and _name isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

process_formdata(valuelist)

widget = <indico.web.forms.widgets.HiddenInputs object>

class indico.web.forms.fields.TextListField (label=None, validators=None, filters=(), de-
scription=u", id=None, default=None, wid-
get=None, render_kw=None, _form=None,
_name=None, _prefix=u", _transla-
tions=None, _meta=None)

Bases: wtforms.fields.simple.TextAreaField

Construct a new field.

Parameters

• label – The label of the field.

• validators – A sequence of validators to call when validate is called.

• filters – A sequence of filters which are run on input data by process.

• description – A description for the field, typically used for help text.

• id – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.

• default – The default value to assign to the field, if no form or object input is provided. May be a callable.

• widget – If provided, overrides the widget used to render the field.

• render_kw (dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.

• _form – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.

• _name – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.

• _prefix – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.

• _translations – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.

• _meta – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.
If _form and _name isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

```python
pre_validate(form)
process_formdata(valuelist)
```

**class** `indico.web.forms.fields.EmailListField`

Construct a new field.

**Parameters**

- **label** – The label of the field.
- **validators** – A sequence of validators to call when validate is called.
- **filters** – A sequence of filters which are run on input data by process.
- **description** – A description for the field, typically used for help text.
- **id** – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.
- **default** – The default value to assign to the field, if no form or object input is provided. May be a callable.
- **widget** – If provided, overrides the widget used to render the field.
- **render_kw**(dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.
- **_form** – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.
- **_name** – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.
- **_prefix** – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.
- **_translations** – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.
- **_meta** – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If _form and _name isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

```python
process_formdata(valuelist)
```

**class** `indico.web.forms.fields.IndicoPasswordField`

Password field which can show or hide the password.

```python
widget = <indico.web.forms.widgets.PasswordWidget object>
```
class indico.web.forms.fields.IndicoStaticTextField(*args, **kwargs)
Bases: wtforms.fields.core.Field

Return an html element with text taken from this field’s value

process_data(data)

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.IndicoTagListField(label=None, validators=None, filters=(), description=u'', id=None, default=None, widget=None, render_kw=None, _form=None, _name=None, _prefix=u'', _translations=None, _meta=None)
Bases: indico.web.forms.fields.simple.HiddenFieldList

Construct a new field.

Parameters

- **label** – The label of the field.
- **validators** – A sequence of validators to call when validate is called.
- **filters** – A sequence of filters which are run on input data by process.
- **description** – A description for the field, typically used for help text.
- **id** – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.
- **default** – The default value to assign to the field, if no form or object input is provided. May be a callable.
- **widget** – If provided, overrides the widget used to render the field.
- **render_kw** (dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.
- **_form** – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.
- **_name** – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.
- **_prefix** – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.
- **_translations** – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.
- **_meta** – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If _form and _name isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.IndicoPalettePickerField(*args, **kwargs)
Bases: indico.web.forms.fields.simple.JSONField

Field allowing user to pick a color from a set of predefined values
CAN_POPULATE = True

pre_validate(form)
process_data(value)
process_formdata(valuelist)

class indico.web.forms.fields.IndicoSinglePalettePickerField(*args, **kwargs)
Bases: indico.web.forms.fields.colors.IndicoPalettePickerField

Like IndicoPalettePickerField but for just a single color.

pre_validate(form)
process_formdata(valuelist)

class indico.web.forms.fields.TimeDeltaField(*args, **kwargs)
Bases: wtforms.fields.core.Field

A field that lets the user select a simple timedelta.

It does not support mixing multiple units, but it is smart enough to switch to a different unit to represent a timedelta that could not be represented otherwise.

Parameters:
units - The available units. Must be a tuple containing any of 'seconds', 'minutes', 'hours' and 'days'. If not specified, ('hours', 'days') is assumed.

best_unit
Return the largest unit that covers the current timedelta

choices

magnitudes = {u'days': 86400, u'hours': 3600, u'minutes': 60, u'seconds': 1}

pre_validate(form)
process_formdata(valuelist)

unit_names = {u'days': u'Days', u'hours': u'Hours', u'minutes': u'Minutes', u'seconds': u'Seconds'}

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.IndicoDateTimeField(*args, **kwargs)
Bases: wtforms.ext.dateutil.fields.DateTimeField

Friendly datetime field that handles timezones and validations.

Important: When the form has a timezone field it must be declared before any IndicoDateTimeField. Otherwise its value is not available in this field resulting in an error during form submission.

earliest_dt
latest_dt

linked_datetime_validator

linked_field

pre_validate(form)
process_formdata(valuelist)

timezone
timezone_field
class indico.web.forms.fields.OccurrencesField(*args, **kwargs)
    Bases: indico.web.forms.fields.simple.JSONField
    A field that lets you select multiple occurrences consisting of a start date/time and a duration.
    CAN_POPULATE = True
    process_formdata(valuelist)

timezone
timezone_field
    widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.IndicoTimezoneSelectField(*args, **kwargs)
    Bases: wtforms.fields.core.SelectField
    process_data(value)

class indico.web.forms.fields.IndicoEnumSelectField(label=None, validators=None, enum=None, sorted=False, only=None, skip=None, none=None, titles=None, keep_enum=True, **kwargs)
    Bases: indico.web.forms.fields.enums._EnumFieldMixin, wtforms.fields.core.SelectFieldBase
    Select field backed by a RichEnum
    iter_choices()
    widget = <wtforms.widgets.core.Select object>

class indico.web.forms.fields.IndicoEnumRadioField(label=None, validators=None, enum=None, sorted=False, only=None, skip=None, none=None, titles=None, keep_enum=True, **kwargs)
    Bases: indico.web.forms.fields.enums.IndicoEnumSelectField
    option_widget = <wtforms.widgets.core.RadioInput object>
    widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.HiddenEnumField(label=None, validators=None, enum=None, only=None, skip=None, none=None, **kwargs)
    Bases: indico.web.forms.fields.enums._EnumFieldMixin, wtforms.fields.simple.HiddenField
    Hidden field that only accepts values from an Enum
    process_formdata(valuelist)

class indico.web.forms.fields.FileField(*args, **kwargs)
    Bases: wtforms.fields.core.Field
    A dropzone field
    default_options = {u'add_remove_links': True, u'handle_flashes': False, u'lightweight': False, u'max_files': 10, u'multiple_files': False}
    process_formdata(valuelist)
`widget = <indico.web.forms.widgets.JinjaWidget object>`

```python
class indico.web.forms.fields.MultiStringField(*args, **kwargs)
    Bases: wtforms.fields.simple.HiddenField

    A field with multiple input text fields.

    Parameters

    • `field` – A tuple (fieldname, title) where the title is used in the placeholder.
    • `uuid_field` – If set, each item will have a UUID assigned and stored in the field specified here.
    • `flat` – If True, the field returns a list of string values instead of dicts. Cannot be combined with `uuid_field`.
    • `unique` – Whether the values should be unique.
    • `sortable` – Whether items should be sortable.

    `pre_validate(form)`
    `process_formdata(valuelist)`
```

```python
class indico.web.forms.fields.MultipleItemsField(*args, **kwargs)
    Bases: wtforms.fields.simple.HiddenField

    A field with multiple items consisting of multiple string values.

    Parameters

    • `fields` – A list of dicts with the following arguments: ‘id’: the unique ID of the field ‘caption’: the title of the column and the placeholder ‘type’: ‘text|number|select’, the type of the field ‘coerce’: callable to convert the value to a python type.

        the type must be convertible back to a string, so usually you just want something like `int` or `float` here.

        In case the type is ‘select’, the property ‘choices’ of the `MultipleItemsField` or the ‘choices’ kwarg needs to be a dict where the key is the ‘id’ of the select field and the value is another dict mapping the option’s id to it caption.
        
        • `uuid_field` – If set, each item will have a UUID assigned and stored in the field specified here. The name specified here may not be in `fields`.
        
        • `uuid_field_opaque` – If set, the `uuid_field` is considered opaque, i.e. it is never touched by this field. This is useful when you subclass the field and use e.g. actual database IDs instead of UUIDs.
        
        • `unique_field` – The name of a field in `fields` that needs to be unique.
        
        • `sortable` – Whether items should be sortable.

    `pre_validate(form)`
    `process_formdata(valuelist)`
```

```python
class indico.web.forms.fields.OverrideMultipleItemsField(*args, **kwargs)
    Bases: wtforms.fields.simple.HiddenField

    A field similar to `MultipleItemsField` which allows the user to override some values.
```

```
Parameters

- **fields** – a list of (fieldname, title) tuples. Should match the fields of the corresponding `MultipleItemsField`.
- **field_data** – the data from the corresponding `MultipleItemsField`.
- **unique_field** – the name of the field which is unique among all rows
- **edit_fields** – a set containing the field names which can be edited

If you decide to use this field, please consider adding support for `uuid_field` here!

**get_overridden_value** *(row, name)*
Utility for the widget to get the entered value for an editable field

**get_row_key** *(row)*
Utility for the widget to get the unique value for a row

**pre_validate** *(form)*

**process_formdata** *(valuelist)*

*widget* = `<indico.web.forms.widgets.JinjaWidget object>`

**class** `indico.web.forms.fields.PrincipalListField` *(*args, **kwargs)*

**Bases:** `wtforms.fields.simple.HiddenField`

A field that lets you select a list Indico user/group (“principal”)

Parameters

- **groups** – If groups should be selectable.
- **allow_networks** – If ip networks should be selectable.
- **allow_emails** – If emails should be allowed.
- **allow_external** – If “search users with no indico account” should be available. Selecting such a user will automatically create a pending user once the form is submitted, even if other fields in the form fail to validate!

**pre_validate** *(form)*

**process_formdata** *(valuelist)*

*widget* = `<indico.web.forms.widgets.JinjaWidget object>`

**class** `indico.web.forms.fields.PrincipalField` *(*args, **kwargs)*

**Bases:** `indico.web.forms.fields.principals.PrincipalListField`

A field that lets you select an Indico user/group (“principal”)

**process_formdata** *(valuelist)*

*widget* = `<indico.web.forms.widgets.JinjaWidget object>`

**class** `indico.web.forms.fields.AccessControlListField` *(*args, **kwargs)*

**Bases:** `indico.web.forms.fields.principals.PrincipalListField`

*widget* = `<indico.web.forms.widgets.JinjaWidget object>`

**class** `indico.web.forms.fields.IndicoQuerySelectMultipleField` *(*args, **kwargs)*

**Bases:** `wtforms.ext.sqlalchemy.fields.QuerySelectMultipleField`

Like the parent, but with a callback that allows you to modify the list

The callback can return a new list or yield items, and you can use it e.g. to sort the list.
data
class indico.web.forms.fields.EditableFileField(*args, **kwargs)
    Bases: indico.web.forms.fields.files.FileField
    A dropzone field that displays its current state and keeps track of deletes.
    process_formdata(valuelist)
    widget = <indico.web.forms.widgets.JinjaWidget object>
class indico.web.forms.fields.IndicoQuerySelectMultipleCheckboxField(*args, **kwargs)
    Bases: indico.web.forms.fields.sqlalchemy.IndicoQuerySelectMultipleField
    option_widget = <wtforms.widgets.core.CheckboxInput object>
    widget = <indico.web.forms.widgets.JinjaWidget object>
class indico.web.forms.fields.IndicoLocationField(*args, **kwargs)
    Bases: indico.web.forms.fields.simple.JSONField
    CAN_POPULATE = True
    process_formdata(valuelist)
    widget = <indico.web.forms.widgets.LocationWidget object>
class indico.web.forms.fields.IndicoMarkdownField(*args, **kwargs)
    Bases: wtforms.fields.simple.TextAreaField
    A Markdown-enhanced text area.
    When using the editor you need to include the markdown JS/CSS bundles and also the MathJax JS bundle (even when using only the editor without MathJax).
    Parameters
        • editor – Whether to use the WMD widget with its live preview
        • mathjax – Whether to use MathJax in the WMD live preview
    widget = <indico.web.forms.widgets.JinjaWidget object>
class indico.web.forms.fields.IndicoDateField(*args, **kwargs)
    Bases: wtforms.ext.dateutil.fields.DateField
    widget = <indico.web.forms.widgets.JinjaWidget object>
class indico.web.forms.fields.IndicoProtectionField(*args, **kwargs)
    Bases: indico.web.forms.fields.enums.IndicoEnumRadioField
    radio_widget = <indico.web.forms.widgets.JinjaWidget object>
    render_protection_message()
    widget = <indico.web.forms.widgets.JinjaWidget object>
class indico.web.forms.fields.IndicoSelectMultipleCheckboxBooleanField (label=None, validators=None, coerce=<type 'unicode'>, choices=None, **kwargs)

Bases: indico.web.forms.fields.simple.IndicoSelectMultipleCheckboxField

iter_choices()

process_formdata (valuelist)

class indico.web.forms.fields.RelativeDeltaField (*args, **kwargs)

Bases: wtforms.fields.core.Field

A field that lets the user select a simple timedelta.

It does not support mixing multiple units, but it is smart enough to switch to a different unit to represent a
timedelta that could not be represented otherwise.

Parameters units – The available units. Must be a tuple containing any any of 'seconds', 'minutes', 'hours' and 'days'. If not specified, ('hours', 'days') is assumed.

choices

magnitudes = {u'days': relativedelta(days=+1), u'hours': relativedelta(hours=+1), u'minutes': relativedelta(minutes=+1)}

pre_validate (form)

process_formdata (valuelist)

split_data

unit_names = {u'days': u'Days', u'hours': u'Hours', u'minutes': u'Minutes', u'months': u'Months'}

widget = <indico.web.forms.widgets.JinjaWidget object>

class indico.web.forms.fields.IndicoWeekDayRepetitionField (*args, **kwargs)

Bases: wtforms.fields.core.Field

Field that lets you select an ordinal day of the week.

WEEK_DAY_NUMBER_CHOICES = ((1, lu'first'), (2, lu'second'), (3, lu'third'), (4, lu'fourth'), (5, lu'last'))

day_number_data

process_formdata (valuelist)

week_day_data

widget = <indico.web.forms.widgets.JinjaWidget object>
class indico.web.forms.fields.IndicoEmailRecipientsField:

Construct a new field.

Parameters

- **label** – The label of the field.
- **validators** – A sequence of validators to call when validate is called.
- **filters** – A sequence of filters which are run on input data by process.
- **description** – A description for the field, typically used for help text.
- **id** – An id to use for the field. A reasonable default is set by the form, and you shouldn’t need to set this manually.
- **default** – The default value to assign to the field, if no form or object input is provided. May be a callable.
- **widget** – If provided, overrides the widget used to render the field.
- **render_kw**(dict) – If provided, a dictionary which provides default keywords that will be given to the widget at render time.
- **_form** – The form holding this field. It is passed by the form itself during construction. You should never pass this value yourself.
- **_name** – The name of this field, passed by the enclosing form during its construction. You should never pass this value yourself.
- **_prefix** – The prefix to prepend to the form name of this field, passed by the enclosing form during construction.
- **_translations** – A translations object providing message translations. Usually passed by the enclosing form during construction. See I18n docs for information on message translations.
- **_meta** – If provided, this is the ‘meta’ instance from the form. You usually don’t pass this yourself.

If **_form** and **_name** isn’t provided, an UnboundField will be returned instead. Call its bind() method with a form instance and a name to construct the field.

process_data(data)

widget = <indico.web.forms.widgets.JinjaWidget object>
What’s New

6.1 Changelog

6.1.1 Version 2.3

Unreleased

Major Features

- Add category roles, which are similar to local groups but within the scope of a category and its subcategories. They can be used for assigning permissions in any of these categories and events within such categories.

Improvements

- Sort survey list by title (#3802)
- Hide “External IDs” field if none are defined (#3857)
- Add LaTeX source export for book of abstracts (#4035, thanks @bpedersen2)
- Tracks can now be categorized in track groups (#4052)
- Program codes for sessions, session blocks, contributions and subcontributions can now be auto-generated (#4026)
- Add draft mode for the contribution list of conference events which hides pages like the contribution list and timetable until the event organizers publish the contribution list. (#4095)
- Add ICS export for information in the user dashboard (#4057)
- Allow data syncing with multipass providers which do not support refreshing identity information
- Show more verbose error when email validation fails during event registration (#4177)
- Add link to external map in room details view (#4146)
• Allow up to 9 digits (instead of 6) before the decimal point in registration fees
• Add button to booking details modal to copy direct link (#4230)
• Do not require new room manager approval when simply shortening a booking (#4214)
• Make root category description/title customizable using the normal category settings form (#4231)
• Added new `LOCAL_GROUPS` setting that can be used to fully disable local groups (#4260)
• Log bulk event category changes in the event log (#4241)
• Add CLI commands to block and unblock users (#3845)
• Show warning when trying to merge a blocked user (#3845)
• Allow importing event role members from a CSV file (#4301)
• Allow optional comment when accepting a pre-booking (#4086)
• Log event restores in event log (#4309)
• Warn about cancelling/rejecting whole recurring bookings instead of just specific occurrences (#4092)
• Add “quick cancel” link to room booking reminder emails (#4324)
• Add visual information and filtering options for participants’ registration status to the contribution list (#4318)

**Bugfixes**

• Hide Book of Abstracts menu item if LaTeX is disabled
• Use a more consistent order when cloning the timetable (#4227)
• Do not show unrelated rooms with similar names when booking room from an event (#4089)
• Stop icons from overlapping in the datetime widget (#4342)
• Fix alignment of materials in events (#4344)

**Internal Changes**

• Make React and SemanticUI usable everywhere (#3955)
• Add `before-regform` template hook (#4171, thanks @giusedb)
• Add `registrations` kwarg to the `event.designer.print_badge_template` signal (#4297, thanks @giusedb)

### 6.1.2 Version 2.2.7

**Unreleased**

**Bugfixes**

• Allow slashes in `roomName` export API
6.1.3 Version 2.2.6

Released on February 27, 2020

Bugfixes

- Fix some email fields (error report contact, agreement cc address) being required even though they should be optional
- Avoid browsers prefilling stored passwords in togradable password fields such as the event access key
- Make sure that tickets are not attached to emails sent to registrants for whom tickets are blocked (#4242)
- Fix event access key prompt not showing when accessing an attachment link (#4255)
- Include event title in OpenGraph metadata (#4288)
- Fix error when viewing abstract with reviews that have no scores
- Update requests and pin idna to avoid installing incompatible dependency versions (#4327)

6.1.4 Version 2.2.5

Released on December 06, 2019

Improvements

- Sort posters in timetable PDF export by board number (#4147, thanks @bpedersen2)
- Use lat/lng field order instead of lng/lat when editing rooms (#4150, thanks @bpedersen2)
- Add additional fields to the contribution csv/xlsx export (authors and board number) (#4148, thanks @bpedersen2)

Bugfixes

- Update the Pillow library to 6.2.1. This fixes an issue where some malformed images could result in high memory usage or slow processing.
- Truncate long speaker names in the timetable instead of hiding them (#4110)
- Fix an issue causing errors when using translations for languages with no plural forms (like Chinese).
- Fix creating rooms without touching the longitude/latitude fields (#4115)
- Fix error in HTTP API when Basic auth headers are present (#4123, thanks @uxmaster)
- Fix incorrect font size in some room booking dropdowns (#4156)
- Add missing email validation in some places (#4158)
- Reject requests containing NUL bytes in the POST data (#4159)
- Fix truncated timetable PDF when using “Print each session on a separate page” in an event where the last timetable entry of the day is a top-level contribution or break (#4134, thanks @bpedersen2)
- Only show public contribution fields in PDF exports (#4165)
- Allow single arrival/departure date in accommodation field (#4164, thanks @bpedersen2)
6.1.5 Version 2.2.4

Released on October 16, 2019

Security fixes

- Fix more places where LaTeX input was not correctly sanitized. While the biggest security impact (reading local files) has already been mitigated when fixing the initial vulnerability in the previous release, it is still strongly recommended to update.

6.1.6 Version 2.2.3

Released on October 08, 2019

Security fixes

- Strip @, +, - and = from the beginning of strings when exporting CSV files to avoid security issues when opening the CSV file in Excel
- Use 027 instead of 000 umask when temporarily changing it to get the current umask
- Fix LaTeX sanitization to prevent malicious users from running unsafe LaTeX commands through specially crafted abstracts or contribution descriptions, which could lead to the disclosure of local file contents

Improvements

- Improve room booking interface on small-screen devices (#4013)
- Add user preference for room owners/manager to select if they want to receive notification emails for their rooms (#4096, #4098)
- Show family name field first in user search dialog (#4099)
- Make date headers clickable in room booking calendar (#4099)
- Show times in room booking log entries (#4099)
- Support disabling server-side LaTeX altogether and hide anything that requires it (such as contribution PDF export or the Book of Abstracts). **LaTeX is now disabled by default, unless the XELATEX_PATH is explicitly set in indico.conf.**

Bugfixes

- Remove 30s timeout from dropzone file uploads
- Fix bug affecting room booking from an event in another timezone (#4072)
- Fix error when commenting on papers (#4081)
- Fix performance issue in conferences with public registration count and a high amount of registrations
- Fix confirmation prompt when disabling conference menu customizations (#4085)
- Fix incorrect days shown as weekend in room booking for some locales
- Fix ACL entries referencing event roles from the old event when cloning an event with event roles in the ACL. Run `indico maint fix-event-role-acls` after updating to fix any affected ACLs (#4090)
• Fix validation issues in coordinates fields when editing rooms (#4103)

6.1.7 Version 2.2.2

Released on August 23, 2019

Bugfixes

• Remove dependency on pyatom, which has vanished from PyPI

6.1.8 Version 2.2.1

Released on August 16, 2019

Improvements

• Make list of event room bookings sortable (#4022)
• Log when a booking is split during editing (#4031)
• Improve “Book” button in multi-day events (#4021)

Bugfixes

• Add missing slash to the template_prefix of the designer module
• Always use HH:MM time format in book-from-event link
• Fix timetable theme when set to “indico weeks view” before 2.2 (#4027)
• Avoid flickering of booking edit details tooltip
• Fix outdated browser check on iOS (#4033)

6.1.9 Version 2.2

Released on August 06, 2019

Major Changes

• Drop support for Internet Explorer 11 and other outdated or discontinued browser versions. Indico shows a warning message when accessed using such a browser. The latest list of supported browsers can be found in the README on GitHub, but generally Indico now supports the last two versions of each major browser (determined at release time), plus the current Firefox ESR.
• Rewrite the room booking frontend to be more straightforward and user-friendly. Check our blog for details.
Improvements

- Rework the event log viewer to be more responsive and not freeze the whole browser when there are thousands of log entries
- Add shortcut to next upcoming event in a category (#3388)
- Make registration period display less confusing (#3359)
- Add edit button to custom conference pages (#3284)
- Support markdown in survey questions (#3366)
- Improve event list in case of long event titles (#3607, thanks @nop33)
- Include event page title in the page’s `<title>` (#3285, thanks @bpedersen2)
- Add option to include subcategories in upcoming events (#3449)
- Allow event managers to override the name format used in the event (#2455)
- Add option to not clone venue/room of an event
- Show territory/country next to the language name (#3968)
- Add more sorting options to book of abstracts (#3429, thanks @bpedersen2)
- Add more formatting options to book of abstracts (#3335, thanks @bpedersen2)
- Improve message when the call for abstracts is scheduled to open but hasn’t started yet
- Make link color handling for LaTeX pdfs configurable (#3283, thanks @bpedersen2)
- Preserve displayed order in contribution exports that do not apply any specific sorting (#4005)
- Add author list button to list of papers (#3978)

Bugfixes

- Fix incorrect order of session blocks inside timetable (#2999)
- Add missing email validation to contribution CSV import (#3568, thanks @Kush22)
- Do not show border after last item in badge designer toolbar (#3607, thanks @nop33)
- Correctly align centered footer links (#3599, thanks @nop33)
- Fix top/right alignment of session bar in event display view (#3599, thanks @nop33)
- Fix error when trying to create a user with a mixed-case email address in the admin area
- Fix event import if a user in the exported data has multiple email addresses and they match different users
- Fix paper reviewers getting notifications even if their type of reviewing has been disabled (#3852)
- Correctly handle merging users in the paper reviewing module (#3895)
- Show correct number of registrations in management area (#3935)
- Fix sorting book of abstracts by board number (#3429, thanks @bpedersen2)
- Enforce survey submission limit (#3256)
- Do not show “Mark as paid” button and checkout link while a transaction is pending (#3361, thanks @driehle)
- Fix 404 error on custom conference pages that do not have any ascii chars in the title (#3998)
- Do not show pending registrants in public participant lists (#4017)
Internal Changes

- Use webpack to build static assets
- Add React+Redux for new frontend modules
- Enable modern ES201x features

6.1.10 Version 2.1.11

Released on October 16, 2019

Security fixes

- Fix more places where LaTeX input was not correctly sanitized. While the biggest security impact (reading local files) has already been mitigated when fixing the initial vulnerability in the previous release, it is still strongly recommended to update.

6.1.11 Version 2.1.10

Released on October 08, 2019

Security fixes

- Strip @, +, – and = from the beginning of strings when exporting CSV files to avoid security issues when opening the CSV file in Excel
- Use 027 instead of 000 umask when temporarily changing it to get the current umask
- Fix LaTeX sanitization to prevent malicious users from running unsafe LaTeX commands through specially crafted abstracts or contribution descriptions, which could lead to the disclosure of local file contents

6.1.12 Version 2.1.9

Released on August 26, 2019

Bugfixes

- Fix bug in calendar view, due to timezones (#3903)
- Remove dependency on pyatom, which has vanished from PyPI (#4045)

6.1.13 Version 2.1.8

Released on March 12, 2019

Improvements

- Add A6 to page size options (#3793)
Bugfixes

- Fix celery/redis dependency issue (#3809)

6.1.14 Version 2.1.7

Released on January 24, 2019

Improvements

- Add setting for the default contribution duration of an event (#3446)
- Add option to copy abstract attachments to contributions when accepting them (#3732)

Bugfixes

- Really fix the oauthlib conflict (was still breaking in some cases)

6.1.15 Version 2.1.6

Released on January 15, 2019

Bugfixes

- Allow adding external users as speakers/chairpersons (#3562)
- Allow adding external users to event ACLs (#3562)
- Pin requests-oauthlib version to avoid dependency conflict

6.1.16 Version 2.1.5

Released on December 06, 2018

Improvements

- Render the reviewing state of papers in the same way as abstracts (#3665)

Bugfixes

- Use correct speaker name when exporting contributions to spreadsheets
- Use friendly IDs in abstract attachment package folder names
- Fix typo in material package subcontribution folder names
- Fix check on whether registering for an event is possible
- Show static text while editing registrations (#3682)
6.1.17 Version 2.1.4

Released on September 25, 2018

Bugfixes

- Let managers download tickets for registrants even if all public ticket downloads are disabled (#3493)
- Do not count deleted registrations when printing tickets from the badge designer page
- Hide “Save answers” in surveys while not logged in
- Fix importing event archives containing registrations with attachments
- Fix display issue in participants table after editing data (#3511)
- Fix errors when booking rooms via API

6.1.18 Version 2.1.3

Released on August 09, 2018

Security fixes

- Only return timetable entries for the current session when updating a session through the timetable (#3474, thanks @glunardi for reporting)
- Prevent session managers/coordinators from modifying certain timetable entries or scheduling contributions not assigned to their session
- Restrict access to timetable entry details to users who are authorized to see them

Improvements

- Improve survey result display (#3486)
- Improve email validation for registrations (#3471)

Bugfixes

- Point to correct day in “edit session timetable” link (#3419)
- Fix error when exporting abstracts with review questions to JSON
- Point the timetable to correct day in the session details
- Fix massive performance issue on the material package page in big events
- Fix error when using the checkin app to mark someone as checked in (#3473, thanks @femtobit)
- Fix error when a session coordinator tries changing the color of a break using the color picker in the balloon’s tooltip

Internal Changes

- Add some new signals and template hooks to the registration module
6.1.19 Version 2.1.2

Released on June 11, 2018

Improvements

• Show email address for non-anonymous survey submissions (#3258)

Bugfixes

• Show question description in survey results (#3383)
• Allow paper managers to submit paper revisions
• Fix error when not providing a URL for privacy policy or terms
• Use consistent order for privacy/terms links in the footer
• Fix cloning of locked events

6.1.20 Version 2.1.1

Released on May 31, 2018

Improvements

• Add a privacy policy page linked from the footer (#1415)
• Terms & Conditions can now link to an external URL
• Show a warning to all admins if Celery is not running or outdated
• Add registration ID placeholder for badges (#3370, thanks @bpedersen2)

Bugfixes

• Fix alignment issue in the “Indico Weeks View” timetable theme (#3367)
• Reset visibility when cloning an event to a different category (#3372)

6.1.21 Version 2.1

Released on May 16, 2018

Major Features

• Add event roles, which are similar to local groups but within the scope of an event. They can be used both for assigning permissions within the event and also for quickly seeing which user has which role (such as “Program Committee” in the event
• Add new Participant Roles (previously called Roles) which now shows each person’s custom event roles and whether they have registered for the event in addition to the the default roles (speaker, chairperson, etc.)
• Add visibility options to custom abstract/contribution fields so they can be restricted to be editable/visible only for event managers or authors/submitters instead of anyone who can see the abstract/contribution

• Provide new interface to import registrations/contributions from a CSV file (#3144)

• Rework how access/permissions are managed. Now all access and management privileges can be assigned from a single place on the protection management page.

**Improvements**

• Allow specifying a default session for a track which will then be used by default when accepting an abstract in that track (#3069)

• Allow marking contribution types as private so they cannot be selected by users submitting an abstract (#3138)

• Add support for boolean (yes/no) and freetext questions in abstract reviewing (#3175)

• Support event cloning with monthly recurrence on the last day of the month (#1580)

• Add support for custom session types (#3189)

• Move poster session flag from session settings to session type settings

• Add contribution cloning within an event (#3207)

• Add option to include the event description in reminder emails (#3157, thanks @bpedersen2)

• Pin default themes to the top for event managers (#3166)

• Add user setting whether to show future events or not by default in a category. Also keep the per-category status in the session (#3233, thanks @bpedersen2)

• Keep page titles in sync with conference menu item titles (#3236)

• Add option to hide an attachment folder in the display areas of an event (#3181, thanks @bpedersen2)

• Improve flower redirect URI generation (#3187, thanks @bpedersen2)

• When blocking a user account, the user will be forcefully logged out in addition to being prevented from logging in

• Show track-related columns in abstract list only if there are tracks defined for the event (#2813)

• Show warning box to inform that reviewer roles do not apply when an event has no tracks (#2919)

• Allow specifying min/max length for registration form text fields (#3193, thanks @bpedersen2)

• Add settings to configure the scale of ‘rating’ questions in paper reviewing

• Show a nicer error message when entering an excessively high base registration fee (#3260)

• Use proper British English for person titles (#3279)

• Add event keywords in meta tags (#3262, thanks @bpedersen2)

• Improve sorting by date fields in the registrant list

• Use the user’s preferred name format in more places

• Add “back to conference” link when viewing a conference timetable using a meeting theme (#3297, thanks @bpedersen2)

• Allow definition lists in places where Markdown or HTML is accepted (#3325)

• Include event date/time in registration emails (#3337)

• Allow div/span/pre with classes when writing raw HTML in CKEditor (#3332, thanks @bpedersen2)
• Sort abstract authors/speakers by last name (#3340)
• Improve machine-readable metadata for events and categories (#3287, thanks @bpedersen2)

Bugfixes

• Fix selecting a person’s title in a different language than English
• Fix display issue in “now happening” (#3278)
• Fix error when displaying the value of an accommodation field in the registrant list and someone has the “no accommodation” option selected (#3272, thanks @bpedersen2)
• Use the ‘Reviewing’ realm when logging actions from the abstract/paper reviewing modules
• Fix error when printing badges/posters with empty static text fields (#3290)
• Fix error when generating a PDF timetable including contribution abstracts (#3289)
• Do not require management access to a category to select a badge template from it as a backside.
• Fix breadcrumb metadata (#3321, thanks @bpedersen2)
• Fix error when accessing certain registration pages without an active registration
• Use event timezone when displaying event log entries (#3354)
• Correctly render most markdown elements when generating a programme PDF (#3351)
• Do not send any emails when trying to approve/reject a registration that is not pending (#3358)

Internal Changes

• Rename Roles in ACL entries to Permissions. This especially affects the can_manage method whose role argument has been renamed to permission (#3057)
• Add new registration_checkin_updated signal that can be used by plugins to perform an action when the checkin state of a registration changes (#3161, thanks @bpedersen2)
• Add new signals that allow plugins to run custom code at the various stages of the RH execution and replace/modify the final response (#3227)
• Add support for building plugin wheels with date/commit-suffixed version numbers (#3232, thanks @driehle)

6.1.22 Version 2.0.3

Released on March 15, 2018

Security fixes

• Do not show contribution information (metadata including title, speakers and a partial description) in the contribution list unless the user has access to a contribution

Improvements

• Show more suitable message when a service request is auto-accepted (#3264)
### 6.1.23 Version 2.0.2

*Released on March 07, 2018*

**Security fixes**

- Update `bleach` to fix an XSS vulnerability

**Improvements**

- Warn when editing a speaker/author would result in duplicate emails

**Bugfixes**

- Take ‘center’ orientation of badge/poster backgrounds into account (#3238, thanks @bpedersen2)
- Fail nicely when trying to register a local account with an already-used email confirmation link (#3250)

### 6.1.24 Version 2.0.1

*Released on February 6, 2018*

**Improvements**

- Add support for admin-only designer placeholders. Such placeholders can be provided by custom plugins and only be used in the designer by Indico admins (#3210)
- Sort contribution types alphabetically
- Add folding indicators when printing foldable badges (#3216)

**Bugfixes**

- Fix LaTeX rendering issue when consecutive lines starting with `[` were present (#3203)
- Do not allow managers to retrieve tickets for registrants for whom ticket access is blocked by a plugin (#3208)
- Log a warning instead of an exception if the Indico version check fails (#3209)
- Wrap long lines in event log entries instead of truncating them
- Properly show message about empty agenda in reminders that have “Include agenda” enabled but an empty timetable
- Fix overly long contribution type names pushing edit/delete buttons outside the visible area (#3215)
- Only apply plugin-imposed ticket download restrictions for tickets, not for normal badges.
- Fix switching between badge sides in IE11 (#3214)
- Do not show poster templates as possible backsides for badges
- Convert alpha-channel transparency to white in PDF backgrounds
- Make number inputs big enough to show 5 digits in chrome
- Sort chairperson list on lecture pages
• Remove whitespace before commas in speaker lists
• Hide author UI for subcontribution speakers (#3222)

### 6.1.25 Version 2.0

*Released on January 12, 2018*

**Improvements**

- Add `author_type` and `is_speaker` fields for persons in the JSON abstract export
- Add legacy redirect for `conferenceTimeTable.py`

**Bugfixes**

- Fix unicode error when searching external users from the “Search Users” dialog
- Fix missing event management menu/layout when creating a material package from the event management area
- Fix error when viewing a contribution with co-authors
- Fix sorting of registration form items not working anymore after moving/disabling some items
- Fix error after updating from 2.0rc1 if there are cached Mako templates
- Fix error when retrieving an image referenced in an abstract fails
- Fix rendering of time pickers in recent Firefox versions (#3194)
- Fix error when trying to use the html serializer with the timetable API
- Fix error when receiving invalid payment events that should be ignored
- Fix last occurrence not being created when cloning events (#3192)
- Fix multiple links in the same line being replaced with the first one when converting abstracts/contributions to PDF (#2816)
- Fix PDF generation when there are links with & in the URL
- Fix incorrect spacing in abstract author/speaker lists (#3205)

### 6.1.26 Version 2.0rc2

*Released on December 8, 2017*

**Improvements**

- Allow changing the reloader used by the dev server (#3150)
Bugfixes

- Do not show borders above/below the message in registration emails unless both the header and body blocks are used (#3151)
- Roll-back the database transaction when an error occurs.
- Fix rendering of the LaTeX error box (#3163)
- Fix “N/A” being displayed in a survey result if 0 is entered in a number field
- Fix “N/A” not being displayed in a survey result if nothing is selected in a multi-choice select field
- Fix error when using target_* placeholders in abstract notification emails for actions other than “Merged” (#3171)
- Show full track title in tooltips on abstract pages
- Show correct review indicators when a reviewer still has to review an abstract in a different track
- Fix unicode error when searching external users in an LDAP backend

Internal Changes

- Remove SCSS_DEBUG_INFO config option.

6.1.27 Version 2.0rc1

Released on November 10, 2017

Improvements

- Hide category field in event creation dialog if there are no subcategories (#3112)
- Remove length limit from registration form field captions (#3119)
- Use semicolons instead of commas as separator when exporting list values (such as multi-select registration form fields) to CSV or Excel (#3060)
- Use custom site title in page title (#3018)
- Allow manually entering dates in datetime fields (#3136)
- Send emails through a celery task. This ensures users do not get an error if the mail server is temporarily unavailable. Sending an email is also retried for a while in case of failure. In case of a persistent failure the email is dumped to the temp directory and can be re-sent manually using the new indico resend_email command (#3121)
- Reject requests containing NUL bytes in the query string (#3142)

Bugfixes

- Do not intercept HTTP exceptions containing a custom response. When raising such exceptions we do not want the default handling but rather send the custom response to the client.
- Do not apply margin for empty root category sidebar (#3116, thanks @nop33)
- Fix alignment of info-grid items on main conference page (#3126)
• Properly align the label of the attachment folder title field
• Fix some rare unicode errors during exception handling/logging
• Clarify messages in session block rescheduling dialogs (#3080)
• Fix event header bar in IE11 (#3135)
• Fix footer on login page (#3132)
• Use correct module name for abstract notification emails in the event log
• Remove linebreaks from email subject in paper review notifications
• Fix extra padding in the CFA roles dialog (#3129)
• Do not show an extra day in timetable management if an event begins before a DST change
• Disable caching when retrieving the list of unscheduled contributions
• Process placeholders in the subject when emailing registrants
• Fix Shibboleth login with non-ascii names (#3143)

Internal Changes

• Add new is_ticket_blocked signal that can be used by plugins to disable ticket downloads for a registration.

6.1.28 Version 2.0a1

Released on October 20, 2017

This is the first release of the 2.0 series, which is an almost complete rewrite of Indico based on a modern software stack and PostgreSQL.
CHAPTER 7

Indices and tables

• genindex
• modindex
8.1 Contact

8.1.1 Website

The official website of Indico is getindico.io, there you can find useful information related to the project.

8.1.2 IRC

We use IRC as our main means of communication among the development team. Get in touch through the official #indico channel on Freenode (irc.freenode.net). It is also accessible through Matrix.

8.1.3 Forum

For more elaborate questions and discussions we encourage you to use our discussion forum.

8.1.4 Issue tracker

We use GitHub issues for specific bug reports and feature requests. Support enquiries are better suited for the IRC channel or the forums.

8.1.5 Twitter

Indico has an official Twitter account, @getindico which is occasionally used for announcements.
indico.core.plugins, 63
indico.modules.attachments, 250
indico.modules.attachments.models.attachments, 250
indico.modules.attachments.models.folders, 252
indico.modules.attachments.models.principals, 255
indico.modules.attachments.operations, 256
indico.modules.attachments.preview, 257
indico.modules.attachments.util, 256
indico.modules.auth, 271
indico.modules.auth.models.identities, 271
indico.modules.auth.models.registration_requests, 271
indico.modules.auth.util, 272
indico.modules.categories, 234
indico.modules.categories.fields, 296
indico.modules.categories.models.categories, 234
indico.modules.categories.models.principals, 237
indico.modules.categories.models.settings, 238
indico.modules.categories.operations, 238
indico.modules.categories.serialize, 239
indico.modules.categories.settings, 240
indico.modules.categories.util, 239
indico.modules.designer, 282
indico.modules.designer.models.images, 282
indico.modules.designer.models.templates, 283
indico.modules.designer.pdf, 284
indico.modules.designer.placeholders, 284
indico.modules.designer.util, 283
indico.modules.events, 103
indico.modules.events.abstracts, 124
indico.modules.events.abstracts.fields, 292
indico.modules.events.abstracts.models.abstracts, 124
indico.modules.events.abstracts.models.call_for_abstracts, 128
indico.modules.events.abstracts.models.comments, 128
indico.modules.events.abstracts.models.email_logs, 129
indico.modules.events.abstracts.models.email_templates, 130
indico.modules.events.abstracts.models.email_template, 130
indico.modules.events.abstracts.models.fields, 131
indico.modules.events.abstracts.models.files, 132
indico.modules.events.abstracts.models.persons, 132
indico.modules.events.abstracts.models.related_tracks, 133
indico.modules.events.abstracts.models.review_questions, 133
indico.modules.events.abstracts.models.review_ratings, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.reviews, 134
indico.modules.events.abstracts.models.roles, 136
indico.modules.events.abstracts.models.settings, 136
indico.modules.events.abstracts.models.settings, 136
indico.modules.events.abstracts.models.settings, 136
indico.modules.events.abstracts.models.settings, 136
indico.modules.events.agreements, 143
indico.modules.events.agreements.models.agreements, 329
indico.modules.events.agreements.placeholders
indico.modules.events.agreements.util,
indico.modules.events.agreements
indico.modules.events.contributions
indico.modules.events.contributions.fields
indico.modules.events.contributions.models.contributions
indico.modules.events.contributions.models.fields
indico.modules.events.contributions.models.persons
indico.modules.events.contributions.models.principals
indico.modules.events.contributions.models.references
indico.modules.events.contributions.models.subcontributions
indico.modules.events.contributions.models.types
indico.modules.events.contributions.operations
indico.modules.events.contributions.util
indico.modules.events.features
indico.modules.events.features.util
indico.modules.events.fields
indico.modules.events.layout
indico.modules.events.layout.models.images
indico.modules.events.layout.models.menu
indico.modules.events.layout.util
indico.modules.events.logs
indico.modules.events.logs.models.entries
indico.modules.events.logs.renderers
indico.modules.events.logs
indico.modules.events.models.events
indico.modules.events.models.persons
indico.modules.events.models.principals
indico.modules.events.models.references
indico.modules.events.models.reviews
indico.modules.events.models.series
indico.modules.events.models.settings
indico.modules.events.models.static_list_links
indico.modules.events.notes
indico.modules.events.notes.models.notes
indico.modules.events.notes.util
indico.modules.events.notes
indico.modules.events.operations
indico.modules.events.papers
indico.modules.events.papers.models.call_for_papers
indico.modules.events.papers.models.comments,
indico.modules.events.papers.models.competences
indico.modules.events.papers.models.files,
indico.modules.events.papers.models.papers
indico.modules.events.papers.models.review_questions
indico.modules.events.papers.models.review_ratings,
indico.modules.events.papers.models.reviews,
indico.modules.events.papers.models.revisions,
indico.modules.events.papers.models.templates
indico.modules.events.papers.models.user_contributions
indico.modules.events.papers.operations
indico.modules.events.papers.util
indico.modules.events.payment
indico.modules.events.payment.models.transactions
indico.modules.events.payment.plugins
indico.modules.events.payment
indico.modules.events.persons
indico.modules.events.persons.operations
indico.modules.events.persons.placeholders
indico.modules.events.models.principals
indico.modules.events.models.references,
indico.modules.events.models.reviews,
indico.modules.events.models.series
indico.modules.events.models.settings,
indico.modules.events.models.static_list_links
330 Python Module Index
indico.modules.rb.models.rooms, 258
indico.modules.rb.models.util, 269
indico.modules.rb.statistics, 270
indico.modules.rb.util, 270
indico.modules.users, 241
indico.modules.users.ext, 250
indico.modules.users.models.affiliations, 245
indico.modules.users.models.emails, 246
indico.modules.users.modelsfavorites, 246
indico.modules.users.models.settings, 247
indico.modules.users.models.suggestions, 246
indico.modules.users.models.users, 241
indico.modules.users.operations, 248
indico.modules.users.util, 249
indico.modules.vc, 277
indico.modules.vc.exceptions, 282
indico.modules.vc.models.vc_rooms, 277
indico.modules.vc.plugins, 280
indico.modules.vc.util, 280
indico.web.forms.fields, 297
Index

A

absolute_download_url (in
  indico.modules.attachments.models.attachments.Attachment
  attribute), 250

Abstract (class in in-
  dico.modules.events.abstracts.models.abstracts),
  124

abstract (indico.modules.events.abstracts.models.comments.AbstractComment
  attribute), 129

abstract (indico.modules.events.abstracts.models.email_logs.AbstractEmailLogEntry
  attribute), 130

abstract (indico.modules.events.abstracts.models.files.AbstractFile
  attribute), 132

abstract (indico.modules.events.abstracts.models.reviews.AbstractReview
  attribute), 135

abstract (indico.modules.events.contributions.models.contributions.Contribution
  attribute), 145

abstract_created (in module in-
  dico.core.signals.event), 69

abstract_deleted (in module in-
  dico.core.signals.event), 69

abstract_id (indico.modules.events.abstracts.models.comments.AbstractComment
  attribute), 129

abstract_id (indico.modules.events.abstracts.models.email_logs.AbstractEmailLogEntry
  attribute), 130

abstract_id (indico.modules.events.abstracts.models.files.AbstractFile
  attribute), 131

abstract_id (indico.modules.events.abstracts.models.files.AbstractField
  attribute), 132

abstract_id (indico.modules.events.abstracts.models.fields.AbstractFieldValue
  attribute), 131

abstract_state_changed (in module in-
  dico.core.signals.event), 69

abstract_title (in
  indico.modules.events.abstracts.settings.BOASortField
  attribute), 142

abstract_updated (in module in-
  dico.core.signals.event), 69

AbstractAction (class in in-
  dico.modules.events.abstracts.models.reviews), 134

AbstractComment (class in in-
  dico.modules.events.abstracts.models.comments), 128

AbstractCommentVisibility (class in in-
  dico.modules.events.abstracts.models.reviews), 128

AbstractEmailLogEntry (class in in-
  dico.modules.events.abstracts.models.email_logs), 129

AbstractEmailTemplate (class in in-
  dico.modules.events.abstracts.models.email_templates), 130

AbstractField (class in in-
  dico.modules.events.abstracts.models.fields), 292

AbstractFieldValue (class in in-
  dico.modules.events.abstracts.models.fields), 131

AbstractIDPlaceholder (class in in-
  dico.modules.events.abstracts.placeholders), 139

AbstractInvitationURLPlaceholder (class in in-
  dico.modules.events.abstracts.placeholders), 139

AbstractPersonLink (class in in-
  dico.modules.events.abstracts.models.persons), 132

AbstractPersonLinkListField (class in in-
  dico.modules.events.abstracts.models.persons), 132

AbstractReview (class in in-
  dico.modules.events.abstracts.models.reviews), 134

AbstractPersonLinkListField (class in in-
  dico.modules.events.abstracts.models.persons), 132

AbstractPublicState (class in in-
  dico.modules.events.abstracts.models.abstracts), 127
AbstractReview (class in indico.modules.events.abstracts.models.reviews), accepted_condition_types (indico.modules.events.abstracts.fields.EmailRuleListField attribute), 268
AbstractReviewingState (class in indico.modules.events.abstracts.models.abstracts), accepted_contrib_type (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 127
AbstractReviewQuestion (class in indico.modules.events.abstracts.models.review_questions), accepted_contrib_type_id (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 125
AbstractReviewRating (class in indico.modules.events.abstracts.models.review_ratings), accepted_on_behalf (indico.modules.events.agreements.models.agreements.AgreementState attribute), 144
AbstractSessionPlaceholder (class in indico.modules.events.abstracts.placeholders), accepted_revision (indico.modules.events.papers.models.papers.Paper attribute), 171
AbstractState (class in indico.modules.events.abstracts.models.abstracts), accepted_track (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 125
AbstractTitlePlaceholder (class in indico.modules.events.abstracts.placeholders), accepted_track_id (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 125
AbstractTrackPlaceholder (class in indico.modules.events.abstracts.placeholders), access_key (indico.modules.attachments.models.attachments.Attachment attribute), 250
AbstractURLPlaceholder (class in indico.modules.events.abstracts.placeholders), access_key (indico.modules.attachments.models.folders.AttachmentFolder attribute), 253
accept (indico.modules.events.abstracts.models.reviews.AbstractAction attribute), 234
accept (indico.modules.events.papers.models.reviews.PaperAction attribute), 145
accept() (indico.modules.events.agreements.models.agreements.Agreement attribute), 104
accept() (indico.modules.events.requests.base.RequestDefinitionBase attribute), 211
accept() (indico.modules.rb.models.revisions.Reservation attribute), 230
accepted (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 127
access_token (indico.modules.oauth.models.tokens.OAuthToken attribute), 253
accepted (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 274
AccessControllListField (class in indico.modules.events.agreements.models.agreements.Agreement attribute), 143
accommodationStats (class in indico.modules.events.agreements.models.agreements.Agreement attribute), 144
acl (indico.modules.attachments.models.attachments.Attachment attribute), 251
acl (indico.modules.attachments.models.folders.AttachmentFolder attribute), 253
accepted (indico.modules.events.registration.models.invitations.InvitationState attribute), 253
acl_entries (indico.modules.attachments.models.attachments.Attachment attribute), 195
accepted (indico.modules.events.requests.models.requests.RequestState attribute), 208
acl_entries (indico.modules.attachments.models.folders.AttachmentFolder attribute), 253
accepted (indico.modules.rb.models.blocked_rooms.BlockedRoom attribute), 263
acl_entries (indico.modules.categories.models.categories.Category attribute), 268
accepted (indico.modules.rb.models.reservations.ReservationState attribute), 268

acl_entries (indico.modules.events.contributions.models.contributions.Contribution attribute), 146
acl_entries (indico.modules.events.models.events.Event attribute), 104
acl_entries (indico.modules.events.sessions.models.sessions.Session fields in module indico.core.signals), 66
acl_entries (indico.modules.events.tracks.models.tracks.Track attribute), 230
acl_event_settings (indico.core.plugins.IndicoPlugin attribute), 258
acl_proxy_class (indico.core.plugins.IndicoPlugin attribute), 63
acl_settings (indico.core.plugins.IndicoPlugin attribute), 222
acl_settings (indico.core.plugins.VCPluginMixin attribute), 280
active_and_answered (indico.modules.events.surveys.models.surveys.SurveyState attribute), 218
active_and_clean (indico.modules.events.models.persons.PersonLinkBase attribute), 218
active_fields (indico.modules.events.sessions.models.sessions.Session attribute), 192
active_fields (indico.modules.events.registration.models.models.items.RegistrationForm attribute), 196
active_fields (indico.modules.events.registration.models.models.items.RegistrationForm attribute), 198
active_registrations (indico.modules.events.registration.models.models.items.RegistrationForm attribute), 192
add_file_date_column (indico.modules.events.statistic.models.static.StaticSite attribute), 232
add_file_date_column (indico.modules.events.surveys.models.surveys.SurveyState attribute), 232
add_file_date_column (indico.modules.events.registerations.models.registrations.RegistrationData attribute), 232
add_principal() (indico.modules.events.settings.EventACLProxy method), 222
add_survey_question() (indico.modules.events.surveys.operations), 222
add_survey_section() (indico.modules.events.surveys.operations), 222
add_survey_text() (indico.modules.events.surveys.operations), 223
add_url_rule() (indico.core.plugins.IndicoPluginBlueprintSetupState method), 65
active_and_answered (indico.core.plugins.IndicoPlugin attribute), 218
active_and_clean (indico.core.plugins.IndicoPlugin attribute), 104
address (indico.modules.events.models.persons.EventPerson attribute), 109
address (indico.modules.events.models.persons.PersonLinkBase attribute), 111
address (indico.modules.events.models.persons.User attribute), 160
adjust_payment_form_data() (indico.modules.events.payment.plugins.PaymentPluginMixin method), 184
add_form_fields (indico.modules.events.abstracts.models.files.AbstractFile attribute), 139
add_form_fields (indico.modules.events.abstracts.models.files.PaperFile attribute), 142
add_form_fields (indico.modules.events.abstracts.models.files.PaperTemplate attribute), 140
add_form_fields (indico.modules.events.models.persons.EventPerson attribute), 109
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.SubmitterLastNamePlaceholder attribute), 202
add_form_fields (indico.modules.events.abstracts.placeholders.SubmitterFirstNamePlaceholder attribute), 180
add_form_fields (indico.modules.events.abstracts.placeholders.TargetSubmitterNamePlaceholder attribute), 141
add_form_fields (indico.modules.events.abstracts.placeholders.TargetSubmitterLastNamePlaceholder attribute), 142
add_form_fields (indico.modules.events.abstracts.placeholders.TargetSubmitterFirstNamePlaceholder attribute), 140
add_form_fields (indico.modules.events.abstracts.placeholders.AbstractURLPlaceholder attribute), 184
add_form_fields (indico.modules.events.abstracts.placeholders.SubmitterLastNo attribute), 140
add_form_fields (indico.modules.events.abstracts.placeholders.SubmitterFirstNo attribute), 140
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.SubmitterLastNo attribute), 140
add_form_fields (indico.modules.events.abstracts.placeholders.SubmitterFirstNo attribute), 140
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
add_form_fields (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder attribute), 196
affiliation (indico.modules.events.registration.models.items.PersonalDataType), 195
affiliation (indico.modules.events.registration.models.items.PersonalDataType), 196
affiliation (indico.modules.users.models.users.User), 242
after_commit (in module indico.core.signals), 66
after_process (in module indico.core.signals), 66
Agreement (class in indico.modules.events.agreements.models.agreements), 143
AgreementLinkPlaceholder (class in indico.modules.events.agreements.placeholders), 145
AgreementState (class in indico.modules.events.agreements.models.agreements), 144
all_emails (indico.modules.users.models.users.User), 242
all_files (indico.modules.attachments.models.attachments.Attachment), 251
all_recipients (indico.modules.events.remainders.models.remainders.EventRemind), 255
allocate_friendly_ids () (class method), 146
allow_access_key (indico.modules.events.models.events.Event), 104
allow_attachments (indico.modules.events.abstracts.models.call_for_abstracts.CallForAbstracts), 128
allow_category_roles (indico.modules.events.tracks.models.principals.TrackPrincipal), 257
allow_category_roles (indico.modules.events.tracks.models.principals.TrackPrincipal), 258
allow_category_roles (indico.modules.events.tracks.models.principals.TrackPrincipal), 259
allow_category_roles (indico.modules.events.tracks.models.principals.TrackPrincipal), 260
allow_category_roles (indico.modules.categories.models.principals.CategoryPrincipal), 231
allow_category_roles (indico.modules.categories.models.principals.CategoryPrincipal), 232
allow_category_roles (indico.modules.categories.models.principals.CategoryPrincipal), 233
allow_category_roles (indico.modules.categories.models.principals.CategoryPrincipal), 234
allow_category_roles (indico.modules.categories.models.principals.CategoryPrincipal), 235
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 237
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 238
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 239
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 240
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 241
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 242
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 243
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 244
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 245
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 246
allow_category_roles (indico.modules.events.models.principals.EventPrincipal), 247
allow_no_access_contact (indico.modules.categories.models.categories.Category), 234
allow_no_access_contact (indico.modules.categories.models.categories.Category), 235
allow_no_access_contact (indico.modules.categories.models.categories.Category), 236
allow_no_access_contact (indico.modules.categories.models.categories.Category), 237
allow_no_access_contact (indico.modules.categories.models.categories.Category), 238
allow_no_access_contact (indico.modules.categories.models.categories.Category), 239
allow_no_access_contact (indico.modules.categories.models.categories.Category), 240
ATTACHMENT_FOLDER_ID_COLUMN (in indico.modules.events.sessions.models.sessions.Session attribute), 211
attachment_id (in indico.modules.attachments.models.attachments.AttachmentFile attribute), 252
attachment_id (in indico.modules.attachments.models.principals.AttachmentPrincipal attribute), 256
ATTACHMENT_STORAGE (built-in variable), 52
attachment_updated (in module indico.core.signals.attachments), 68
AttachmentFile (class in indico.modules.attachments.models.attachments), 252
AttachmentFolder (class in indico.modules.attachments.models.folders), 252
AttachmentFolderPrincipal (class in indico.modules.attachments.models.principals), 255
AttachmentPrincipal (class in indico.modules.attachments.models.principals), 255
attachments (indico.modules.attachments.models.folders.AttachmentFolder attribute), 253
AttachmentType (class in indico.modules.attachments.models.attachments), 252
attr (indico.modules.events.settings.EventSettingProperty attribute), 123, 180
attribute (indico.modules.rb.models.room_attributes.RoomAttributeAssociation attribute), 261
attribute_id (indico.modules.rb.models.room_attributes.RoomAttributeAssociation attribute), 261
attributes (indico.modules.rb.models.rooms.Room attribute), 258
AUTH_PROVIDERS (built-in variable), 46
author_type (indico.modules.events.abstracts.models.persons.AbstractPersonLink attribute), 132
author_type (indico.modules.events.contributions.models.persons.ContributionPersonLink attribute), 150
author_type (indico.modules.events.contributions.models.persons.SubContributionPersonLink attribute), 151
AUTHORS_SPEAKERS_DISPLAY_ORDER_ATTR (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 125
AUTHORS_SPEAKERS_DISPLAY_ORDER_ATTR (indico.modules.events.models.persons.AuthorsSpeakersMixin attribute), 109
AuthorsSpeakersMixin (class in indico.modules.events.models.persons), 109
AuthorType (class in indico.modules.events.contributions.models.persons), 109
available_equipment (in indico.modules.rb.models.rooms.Room attribute), 258
background_color (in indico.modules.events.sessions.models.sessions.Session attribute), 211
background_color (in indico.modules.events.timetable.models.breaks.Break attribute), 224
background_image (in indico.modules.designer.models.templates.DesignerTemplate attribute), 283
background_image_id (in indico.modules.designer.models.templates.DesignerTemplate attribute), 283
background_position (in indico.modules.designer.pdf.TplData attribute), 284
backside_template (in indico.modules.designer.models.templates.DesignerTemplate attribute), 283
base_price (indico.modules.events.registration.models.forms.RegistrationForm attribute), 192
base_price (indico.modules.events.registration.models.registrations.Registration attribute), 186
before_process (in module indico.core.signals.rh), 74
belongsTo() (in indico.modules.events.agreements.models.agreements.Agreement attribute), 143
best_unit (indico.web.forms.fields.TimeDeltaField attribute), 302
billable_data (in indico.modules.events.registration.models.registrations.Registration attribute), 186
blocks (indico.modules.rb.models.blockings.Blocking attribute), 279
blocked_rooms (in indico.modules.rb.models.blockings.Blocking attribute), 262
Index 339

blocked_rooms (in dico.modules.rb.models.rooms.Room attribute), 258

BlockedRoom (class in in dico.modules.rb.models.blocked_rooms), 262

BlockedRoomState (class in in dico.modules.rb.models.blocked_rooms), 263

Blocking (class in in dico.modules.rb.models.blockings), 262

blocking_id (indico.modules.rb.models.blocked_rooms, BlockedRoom, attribute), 263

blocking_id (indico.modules.rb.models.blocking_principals, BlockingPrincipal, attribute), 263

BlockingPrincipal (class in in dico.modules.rb.models.blocking_principals), 263

blocks (indico.modules.events.sessions.models.sessions.Session attribute), 211

BOACorrespondingAuthorType (class in in dico.modules.events.abstracts.settings), 142

BOALinkFormat (class in in dico.modules.events.abstracts.settings), 142

board_number (indico.modules.events.abstracts.models.contributions.Contribution, attribute), 146

board_number (indico.modules.events.abstracts.models.email_logs.AbstractEmailLogEntry, attribute), 130

board_number (indico.modules.events.abstracts.models.email_templates.AbstractEmailTemplate, attribute), 130

BOASortField (class in in dico.modules.events.abstracts.settings), 142

body (indico.modules.events.abstracts.models.email_logs.AbstractEmailLogEntry, attribute), 167

body (indico.modules.events.abstracts.models.email_templates.AbstractEmailTemplate, attribute), 167

bookable_hours (in dico.modules.rb.models.rooms.Room attribute), 258

BookableHours (class in in dico.modules.rb.models.room_bookable_hours), 261

booked_for_id (indico.modules.rb.models.reservations.Reservation, attribute), 265

booked_for_name (in dico.modules.rb.models.reservations.Reservation attribute), 265

booked_for_user (in dico.modules.rb.models.reservations.Reservation attribute), 265

booking_created (in module indico.core.signals.rb), 74

booking_deleted (in module indico.core.signals.rb), 74

booking_limit_days (in dico.modules.rb.models.rooms.Room attribute), 258

booking_occurrence_state_changed (in module indico.core.signals.rb), 74

booking_occurrence_state_changed (in module indico.core.signals.rb), 74

booking_reason (in dico.modules.rb.models.reservations.Reservation attribute), 266

booking_state_changed (in module indico.core.signals.rb), 74

both (indico.modules.events.abstracts.models.abstracts.EditTrackMode attribute), 128

bottom_left_latitude (in dico.modules.rb.models.map_areas.MapArea attribute), 264

bottom_right_latitude (in dico.modules.rb.models.map_areas.MapArea attribute), 264

Break (class in indico.modules.events.timetable.models.breaks), 224

BREAK (indico.modules.events.timetable.models.entries.TimetableEntryType attribute), 226

break_id (indico.modules.events.timetable.models.entries.TimetableEntry attribute), 128

build_def_template() (in module dico.modules.abs.models.absutil), 137

build_menu_entry_name() (in module indico.modules.abs.models.absutil), 228

build_note_api_data() (in module indico.modules.abs.models.absutil), 228

build_note_legacy_api_data() (in module indico.modules.abs.models.absutil), 228

build_registration_api_data() (in module indico.modules.abs.models.absutil), 228

build_rooms_spritesheet() (in module indico.modules.rb.stat), 270

build_user_search_query() (in module indico.modules.abs.models.absutil), 249

building (in dico.modules.rb.models.rooms.Room attribute), 258

C

CACHE_BACKEND (built-in variable), 46

CACHE_DIR (built-in variable), 50

calculate_price() (in dico.modules.abs.models.absutil), 190

calculate_rooms_bookable_time() (in module indico.modules.rb.stat), 270

calculate_rooms_booked_time() (in module indico.modules.rb.stat), 270
calculate_rooms_occupancy() (in module dico.modules.rb.statistics), 270

call_for_proposals_attr (in module dico.modules.eventsabstracts.models.abstracts.Abstract), 125
call_for_proposals_attr (in module dico.modules.eventsmodels.reviews.ProposalMixin), 125
call_for_proposals_attr (in module dico.modules.events.papers.models.papers.Paper), 171
call_for_proposals_attr (in module dico.modules.events.abstracts.models.abstracts.Abstract), 125
call_for_proposals_attr (in module dico.modules.events.abstracts.models.abstracts.Abstract), 114
can_access (in module indico.core.signals.acl), 67
can_access (in module dico.modules.rb.models.reservations.Reservation), 266
can_access (in module indico.core.signals.acl), 125
can_access (in module dico.modules.rb.models.rooms.Room), 258
can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_access can_acce...
can_swap_entry()  (in module dico.modules.events.timetable.operations), 227

can_view()  (in module dico.modules.attachments.models.folders.AttachmentFolder), 267

can_view()  (in module dico.modules.events.abstracts.models.abstracts.Abstract), 129

can_view()  (in module dico.modules.events.papers.models.reviews.PaperReviewComment), 135

can_view()  (in module dico.modules.events.timetable.models.entries.TimetableEntry), 169

can_view()  (in module dico.modules.events.papers.models.reviews.PaperReview), 104

can_view()  (in module dico.modules.events.timetable.models.entries.TimetableEntry), 54

can_view()  (in module dico.modules.events.payment.plugins.PaymentPluginMixin), 225

can_withdraw()  (in module dico.modules.categories.models.principals.CategoryPrincipal), 125

cancel()  (in module dico.modules.events.payment.models.transactions.TransactionStatus), 183

cancel()  (in module dico.modules.rb.models.reservation_occurrences.ReservationOccurrenceState), 238

cancel()  (in module dico.modules.rb.models.reservations.ReservationLink), 269

cancel()  (in module dico.modules.rb.models.reservations.Reservation), 104

cancelled()  (in module dico.modules.events.payment.models.transactions.TransactionStatus), 183

cancelled()  (in module dico.modules.rb.models.reservation_occurrences.ReservationOccurrenceState), 269

cancelled()  (in module dico.modules.rb.models.reservations.ReservationState), 246

candidate_contrib_types  (in module dico.modules.events.abstracts.models.abstracts.Abstract), 125

candidate_tracks  (in module dico.modules.events.abstracts.models.abstracts.Abstract), 125

capacity  (in module dico.modules.rb.models.rooms.Room), 259

Category  (class in module dico.modules.categories.models.categories), 234

category  (in module dico.core.plugins.IndicoPlugin), 63

category  (in module dico.modules.categories.models.principals.AttachmentFolderPrincipal), 112

category  (in module dico.modules.categories.models.principals.CategoryPrincipal), 237

category  (in module dico.modules.categories.models.principals.AttachmentFolderPrincipal), 255

category  (in module dico.modules.categories.models.principals.AttachmentFolderPrincipal), 266

category  (in module dico.modules.categories.models.principals.AttachmentFolderPrincipal), 280

category  (in module dico.modules.categories.models.principals.AttachmentFolderPrincipal), 283

category  (in module dico.modules.rb.models.reservations.Reservation), 231

category  (in module dico.modules.events.tracks.models.principals.TrackPrincipal), 143

Index
create_judgment_endpoint (in \texttt{dico.modules.events.abstracts.models.abstracts.Abstract} attribute), 125
create_judgment_endpoint (in \texttt{dico.modules.events.models.reviews.ProposalMixin} attribute), 115
create_judgment_endpoint (in \texttt{dico.modules.events.papers.models.papers.Paper} attribute), 171
create_manager_form() (in \texttt{dico.modules.events.requests.base.RequestDefinitionBase} class method), 209
create_mock_abstract() (in module \texttt{dico.modules.events.abstracts.util}), 137
create_next() (in \texttt{dico.modules.events.payment.models.transactions.PaymentTransaction} class method), 182
create_occurrences() (in \texttt{dico.modules.rb.models.reservations.Reservation} method), 266
create_paper_revision() (in module \texttt{dico.modules.events.papers.operations}), 178
create_paper_template() (in module \texttt{dico.modules.events.papers.operations}), 178
create_personal_data_fields() (in module \texttt{dico.modules.events.registration.util}), 200
create_reference_type() (in module \texttt{dico.modules.events.operations}), 119
create_registration() (in module \texttt{dico.modules.events.registration.util}), 200
create_review() (in module \texttt{dico.modules.events.papers.operations}), 178
create_review_endpoint (in \texttt{dico.modules.events.abstracts.models.abstracts.Abstract} attribute), 125
create_review_endpoint (in \texttt{dico.modules.events.models.reviews.ProposalMixin} attribute), 115
create_review_endpoint (in \texttt{dico.modules.events.papers.models.papers.Paper} attribute), 171
create_reviewing_question() (in module \texttt{dico.modules.events.operations}), 119
create_revision() (in \texttt{dico.modules.events.notes.models.notes.EventNote} method), 165
create_room() (in \texttt{dico.modules.vc.plugins.VCPluginMixin} method), 280
create_series() (in \texttt{dico.modules.rb.models.reservation_occurrences.ReservationOccurrence} class method), 269
create_series_for_reservation() (in \texttt{dico.modules.rb.models.reservation_occurrences.ReservationOccurrence} class method), 269
create_session() (in module \texttt{dico.modules.events.sessions.operations}), 215
create_session_block() (in module \texttt{dico.modules.events.sessions.operations}), 215
create_session_block_entry() (in module \texttt{dico.modules.events.timetable.operations}), 227
create_subcontribution() (in module \texttt{dico.modules.events.contributions.operations}), 227
create_timetable_entry() (in module \texttt{dico.modules.events.timetable.operations}), 155
create_untrusted_persons (in \texttt{dico.modules.events.abstracts.fields.AbstractPersonLinkListField} attribute), 292
create_untrusted_persons (in \texttt{dico.modules.events.fields.EventPersonListField} attribute), 291
create_untrusted_persons (in \texttt{dico.modules.events.abstracts.models.abstracts.Abstract} attribute), 292
create_untrusted_persons (in \texttt{dico.modules.events.models.reviews.ProposalMixin} attribute), 227
created (in module \texttt{indico.modules.vc.models.vc_rooms.VCRoomStatus} attribute), 279
created_by_id (in \texttt{dico.modules.events.requests.models.requests.Request} attribute), 207
created_by_id (in \texttt{dico.modules.rb.models.blockings.Blocking} attribute), 262
created_by_id (in \texttt{dico.modules.rb.models.reservations.Reservation} attribute), 266
created_by_id (in \texttt{dico.modules.rb.models.reservation_occurrences.ReservationOccurrence} attribute), 279
created_by_id (in \texttt{dico.modules.vc.plugins.VCPluginMixin} method), 280
created_by_id (in \texttt{dico.modules.rb.models.blockings.Blocking} attribute), 262
created_by_id (in \texttt{dico.modules.rb.models.reservations.Reservation} attribute), 266
created_by_id (in-
default_event_themes (in dico.modules.categories.models.categories.Category attribute), 234
default_list_config (in dico.modules.events.util.ListGeneratorBase attribute), 119
DEFAULT_LOCALE (built-in variable), 54
default_options (indico.web.forms.fields.FileField attribute), 303
default_page (indico.modules.events.models.events.Event attribute), 104
default_page_id (in dico.modules.events.models.events.Event attribute), 104
default_protection_mode (in dico.modules.rb.models.rooms.Room attribute), 259
default_redirect_uri (in dico.modules.oauth.models.applications.OAuthApplication attribute), 273
default_render_mode (in dico.modules.categories.models.categories.Category attribute), 234
default_render_mode (in dico.modules.events.abstracts.models.abstracts.Abstract attribute), 125
default_render_mode (in dico.modules.events.contributions.models.contributions.Contribution attribute), 146
default_render_mode (in dico.modules.events.contributions.models.subcontributions.SubContribution attribute), 153
default_render_mode (in dico.modules.events.models.events.Event attribute), 105
default_render_mode (in dico.modules.events.papers.models.reviews.PaperReview attribute), 174
default_render_mode (in dico.modules.events.papers.models.revisions.PaperRevision attribute), 176
default_render_mode (in dico.modules.events.sessions.models.sessions.Session attribute), 211
default_render_mode (in dico.modules.events.surveys.models.items.SurveyItem attribute), 219
default_render_mode (in dico.modules.events.timetable.models.breaks.Break attribute), 224
default_render_mode (in dico.modules.events.tracks.models.tracks.Track attribute), 230
default_scopes (in dico.modules.oauth.models.applications.OAuthApplication attribute), 273
default_session (in dico.modules.events.tracks.models.tracks.Track attribute), 230
default_session_id (in dico.modules.events.tracks.models.tracks.Track attribute), 230
default_settings (in dico.core.plugins.IndicoPlugin attribute), 63
default_settings (in dico.modules.events.payment.plugins.PaymentPluginMixin attribute), 184
default_settings (in dico.modules.vc.plugins.VCPluginMixin attribute), 280
default_sort_alpha (in dico.modules.events.abstracts.fields.AbstractPersonLinkListField attribute), 292
default_sort_alpha (in dico.modules.events.fields.PersonLinkListFieldBase attribute), 291
default_ticket_template (in dico.modules.events.abstracts.models.reviews.AbstractReview attribute), 135
default_ticket_template (in dico.modules.categories.models.categories.Category attribute), 234
default_ticket_template_id (in dico.modules.categories.models.categories.Category attribute), 235
default_ticket_template_of (in dico.modules.designer.models.templates.DesignerTemplate attribute), 283
DEFAULT_TIMEZONE (built-in variable), 54
default_user_settings (in dico.core.plugins.IndicoPlugin attribute), 63
definition (indico.modules.events.agreements.models.agreements.Agreement attribute), 143
definition (indico.modules.events.requests.models.requests.Request attribute), 207
delete() (in dico.modules.categories.settings.CategorySettingsProxy method), 240
delete() (indico.modules.events.models.events.Event method), 105
delete() (indico.modules.events.notes.models.notes.EventNote method), 165
delete() (indico.modules.events.settings.EventSettingsProxy method), 123, 180

description (indico.modules.designer.placeholders.RegistrationFullNamePlaceholderB, attribute), 285
description (indico.modules.designer.placeholders.RegistrationFullNamePlaceholderC, attribute), 285
description (indico.modules.designer.placeholders.RegistrationFullNamePlaceholderD, attribute), 286
description (indico.modules.designer.placeholders.RegistrationLastNamePlaceholder, attribute), 141
description (indico.modules.designer.placeholders.RegistrationPositionPlaceholder, attribute), 141
description (indico.modules.designer.placeholders.RegistrationTicketQRPlaceholder, attribute), 145
description (indico.modules.events.abstracts.models.review_questions.AbstractReviewQuestion, attribute), 133
description (indico.modules.events.abstracts.placeholders.AbstractIDPlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.AbstractInvitationURLPlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.AbstractSessionPlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.AbstractTitlePlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.AbstractTrackPlaceholder, attribute), 140
description (indico.modules.events.abstracts.placeholders.AbstractURLPlaceholder, attribute), 140
description (indico.modules.events.abstracts.placeholders.CoAuthorsPlaceholder, attribute), 140
description (indico.modules.events.abstracts.placeholders.ContributionURLPlaceholder, attribute), 140
description (indico.modules.events.abstracts.placeholders.EventURLPlaceholder, attribute), 140
description (indico.modules.events.abstracts.placeholders.JudgmentCommentPlaceholder, attribute), 140
description (indico.modules.events.abstracts.placeholders.SubmitterFirstNamePlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.SubmitterLastNamePlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.SubmitterNamePlaceholder, attribute), 139
description (indico.modules.events.abstracts.placeholders.SubmitterTitlePlaceholder, attribute), 140

description (indico.modules.events.registration.placeholders.invitations.InvitationLinkPlaceholder, attribute), 185
description (indico.modules.events.registration.placeholders.invitations.FirstNamePlaceholder, attribute), 185
description (indico.modules.events.registration.models.items.RegistrationFormSection, attribute), 185
description (indico.modules.events.registration.models.items.RegistrationFormPersonalDataSection, attribute), 185
description (indico.modules.events.persons.placeholders.RegisterLinkPlaceholder, attribute), 185
description (indico.modules.events.persons.placeholders.LastNamePlaceholder, attribute), 185
description (indico.modules.events.persons.placeholders.FirstNamePlaceholder, attribute), 185
description (indico.modules.events.persons.placeholders.EventTitlePlaceholder, attribute), 185
description (indico.modules.events.persons.placeholders.EventLinkPlaceholder, attribute), 185
description (indico.modules.events.persons.placeholders.EmailPlaceholder, attribute), 185
description (indico.modules.events.papers.models.review_questions.PaperReviewQuestion, attribute), 177
description (indico.modules.events.CONTRIBUTION_TYPES, attribute), 185
description (indico.modules.events.contributions.models.fields.ContributionField, attribute), 185
description (indico.modules.events.agreements.placeholders.PersonNamePlaceholder, attribute), 185
description (indico.modules.events.agreements.placeholders.AgreementLinkPlaceholder, attribute), 185
description (indico.modules.events.abstracts.placeholders.TargetSubmitterNamePlaceholder, attribute), 185
description (indico.modules.events.abstracts.placeholders.TargetSubmitterLastNamePlaceholder, attribute), 185
description (indico.modules.events.abstracts.placeholders.TargetSubmitterFirstNamePlaceholder, attribute), 185
description (indico.modules.events.abstracts.placeholders.TargetAbstractIDPlaceholder, attribute), 185
Indico Documentation, Release 2.3-dev

description (indico.modules.events.registration.placeholders.registrations.FirstNamePlaceholder attribute), 202
description (indico.modules.events.registration.placeholders.registrations.LastNamePlaceholder attribute), 202
description (indico.modules.events.registration.placeholders.registrations.LinkPlaceholder attribute), 203
description (indico.modules.events.registration.models.forms.RegistrationForm attribute), 203
description (indico.modules.events.pdfutils.models.pdfutils.CeleryCheck disabled), 259
description (indico.modules.users.models.users.PersonMixin attribute), 242
description (indico.modules.events.registration.models.registrations.Registration attribute), 186
description (dico.modules.events.abstracts.models.persons.AbstractPersonLink attribute), 132
description (dico.modules.events.contributions.models.persons.ContributionPersonLink attribute), 150
description (dico.modules.events.contributions.models.persons.SubContributionPersonLink attribute), 151
description (dico.modules.events.models.persons.EventPersonLink attribute), 110
description (dico.modules.events.models.persons.PersonLinkBase attribute), 111
description (dico.modules.events.models.persons.SessionBlockPersonLink attribute), 214
description (dico.modules.categories.models.categories.Category attribute), 203
description (dico.modules.categories.models.categories.EventMessageMode attribute), 203
description (dico.modules.categories.models.categories.EventMessageMode attribute), 111
description (dico.modules.categories.models.categories.EventMessageMode attribute), 235
description (dico.modules.categories.models.categories.EventMessageMode attribute), 146
description (dico.modules.categories.models.categories.EventMessageMode attribute), 152
description (dico.modules.categories.models.categories.EventMessageMode attribute), 235
description (dico.modules.categories.models.categories.EventMessageMode attribute), 111
description (dico.modules.categories.models.categories.EventMessageMode attribute), 214
description (dico.modules.categories.models.categories.EventMessageMode attribute), 259

disabled (dico.modules.categories.models.categories.EventMessageMode attribute), 237
disabled (dico.modules.categories.models.categories.EventMessageMode attribute), 193

disallowed_protection_modes (indico.modules.events.contributions.models.contributions.ContributionPrincipal attribute), 146
disallowed_protection_modes (indico.modules.events.contributions.models.contributions.ContributionPrincipal attribute), 152
disallowed_protection_modes (indico.modules.events.contributions.models.contributions.ContributionPrincipal attribute), 214
disallowed_protection_modes (indico.modules.events.contributions.models.contributions.ContributionPrincipal attribute), 105

disabled (dico.modules.categories.models.categories.EventMessageMode attribute), 235
disallowed_protection_modes (dico.modules.categories.models.categories.EventMessageMode attribute), 259
disallowed_protection_modes (dico.modules.rb.models.rooms.Room attribute), 259
disallowed_protection_modes (dico.modules.rb.models.rooms.Room attribute), 211

Index 353
Indico Documentation, Release 2.3-dev

attribute), 251
download_url (indico.modules.designer.models.images.DesignerImageFile.

DownloadUrl (in module indico.core.signals.users), 282
dr (indico.modules.users.models.users.UserTitle attribute), 245
duplicate (indico.modules.events.abstracts.models.abstracts.Abstract.
duplicate attribute), 127
duplicate (indico.modules.events.abstracts.models.abstracts.Abstract.
duplicate attribute), 128
duplicate_of (indico.modules.events.abstracts.models.abstracts.

duplicate attribute), 125
duplicate_of_id (indico.modules.events.abstracts.models.abstracts.

duplicate attribute), 128
duration (indico.modules.events.contributions.models.contributions.
duration attribute), 146
duration (indico.modules.events.contributions.models.contributions.
duration attribute), 153
duration (indico.modules.events.contributions.models.events.

duration attribute), 105
duration (indico.modules.events.sessions.models.blocks.

duration attribute), 213
duration (indico.modules.events.timetable.models.breaks.

duration attribute), 224
duration (indico.modules.events.timetable.models.entries.

duration attribute), 225
duration (indico.modules.events.timetable.models.reschedule.

duration attribute), 229
duration_display (indico.modules.events.contributions.models.

duration attribute), 146
duration_poster (indico.modules.events.contributions.models.

duration attribute), 146
earliest_dt (indico.web.forms.fields.IndicoDateTimeField.
earliest_dt attribute), 302
edit_comment_endpoint (indico.modules.events.abstracts.models.
edit_comment_endpoint attribute), 125
edit_comment_endpoint (indico.modules.events.models.reviews.
edit_comment_endpoint attribute), 115
edit_comment_endpoint (indico.modules.events.abstracts.models.
edit_comment_endpoint attribute), 171
edit_logs (indico.modules.rb.models.reservations.
edit_logs attribute), 266
edit_review_endpoint (indico.modules.events.abstracts.models.
edit_review_endpoint attribute), 125
edit_review_endpoint (indico.modules.events.abstracts.models.
edit_review_endpoint attribute), 130
event_id (indico.modules.events.logs.models.entries.EventLogEntry), 162
event_id (indico.modules.events.models.persons.EventPerson), 235
event_id (indico.modules.events.models.persons.EventPersonLink), 110
event_id (indico.modules.events.models.principals.EventPrincipal), 243
event_id (indico.modules.events.models.principals.EventPrincipal), 112
event_id (indico.modules.events.models.references.EventReference), 230
event_id (indico.modules.events.models.settings.EventSetting), 124, 181
event_id (indico.modules.events.models.static_list_links.StaticListLink), 256
event_id (indico.modules.events.models.static_list_links.StaticListLink), 118
event_id (indico.modules.events.models.static_list_links.StaticListLink), 117
event_id (indico.modules.events.models.static_list_links.StaticListLink), 233
event_id (indico.modules.events.models.static_list_links.StaticListLink), 217
event_id (indico.modules.events.models.static_list_links.StaticListLink), 255
event_id (indico.modules.events.models.static_list_links.StaticListLink), 117
event_id (indico.modules.events.registration.models.forms.RegistrationForm), 214
event_id (indico.modules.events.registration.models.registrations.Registration), 231
event_id (indico.modules.events.registration.models.registrations.Registration), 186
event_id (indico.modules.events.registrations.models.registrations.Registration), 206
event_id (indico.modules.events.requests.models.requests.Request), 208
event_id (indico.modules.events.sessions.models.sessions.Session), 211
event_id (indico.modules.events.static.models.static.StaticSite), 233
event_id (indico.modules.events.surveys.models.surveys.Survey), 237
event_id (indico.modules.events.timetable.models.entries.TimetableEntry), 225
event_id (indico.modules.events.tracks.models.tracks.Track), 230
event_id (indico.modules.events.rb.models.reservations.ReservationLink), 267
event_id (indico.modules.rb.models.blocking_principals.BlockingPrincipal), 243
event_id (indico.modules.rb.models.blocking_principals.BlockingPrincipal), 112
event_id (indico.modules.vc.models.vc_rooms.VCRoomEventAssociation), 278
event_log_entries (indico.modules.users.models.users.User), 243
event_message (indico.modules.categories.models.categories.Category), 235

Index 357
attribute), 231

event_role_id (in-
dico.modules.rb.models.blocking_principals.BlockingPrincipal attribute), 263
event_roles (indico.modules.events.abstracts.fields.TrackRoleField attribute), 294
event_settings (indico.core.plugins.IndicoPlugin attribute), 63
event_settings_converters (indico.core.plugins.IndicoPlugin attribute), 63
event_settings_form (indico.modules.events.payment.plugins.PaymentPluginMixin attribute), 184
event_start_delta (in-
dico.modules.events.reminders.models.reminders.EventReminder attribute), 206

EventACLProxy (class in in-
dico.modules.events.settings), 122, 179
EventDatesPlaceholder (class in in-
dico.modules.designer.placeholders), 284
EventDescriptionPlaceholder (class in in-
dico.modules.designer.placeholders), 284
EventLinkPlaceholder (class in in-
dico.modules.events.persons.placeholders), 185
EventLogEntry (class in in-
dico.modules.events.logs.models.entries), 162
EventLogKind (class in in-
dico.modules.events.logs.models.entries), 163
EventLogRealm (class in in-
dico.modules.events.logs.models.entries), 163
EventLogRendererBase (class in in-
dico.modules.events.logs.renderers), 163
EventMessageMode (class in in-
dico.modules.categories.models.categories), 237
EventNote (class in in-
dico.modules.events.notes.models.notes), 165
EventNoteRevision (class in in-
dico.modules.events.notes.models.notes), 167
EventOrgTextPlaceholder (class in in-
dico.modules.designer.placeholders), 285
EventPage (class in in-
dico.modules.events.layout.models.menu), 158
EventPerson (class in in-
dico.modules.events.models.persons), 109
EventPersonListField (class in in-
dico.modules.events.fields), 291
EventPrincipal (class in in-
dico.modules.events.models.persons), 110
EventReference (class in in-
dico.modules.events.models.references), 112
EventReminder (class in in-
dico.modules.events.reminders.models.reminders.EventReminder), 206
EventRoomPlaceholder (class in in-
dico.modules.designer.placeholders), 288
Events_backref_name (in-
dico.modules.attachments.models.folders.AttachmentFolder attribute), 253
EventSpeakersPlaceholder (class in in-
dico.modules.events.registration.placeholders.registrations), 202
EventSeries (class in in-
dico.modules.events.models.series), 116
EventSetting (class in in-
dico.modules.events.models.settings), 116
EventSettingPrincipal (class in in-
dico.modules.events.models.persons), 116
EventSettingProperty (class in in-
dico.modules.events.models.settings), 123, 180
EventSettingsMixin (class in in-
dico.modules.events.models.settings), 117
EventSettingsProxy (class in in-
dico.modules.events.models.settings), 123, 180
EventSpeakersPlaceholder (class in in-
dico.modules.designer.placeholders), 288
EventTitlePlaceholder (class in in-
dico.modules.events.models.categories.models.categories), 237
EventTitlePlaceholder (class in in-
dico.modules.events.models.persons.placeholders), 185
EventTitlePlaceholder (class in in-
dico.modules.events.registration.placeholders.registrations), 202
EventType (class in in-
friendly_name
(dico.modules.vc.plugins.VCPluginMixin attribute), 280
full_access (dico.modules.categories.models.principals.CategoryPrincipal attribute), 238
full_access (dico.modules.events.contributions.models.principals.ContributionPrincipal attribute), 152
full_access (dico.modules.events.models.principals.EventPrincipal attribute), 112
full_access (dico.modules.events.sessions.models.principals.SessionPrincipal attribute), 214
full_access (dico.modules.events.tracks.models.principals.TrackPrincipal attribute), 231
full_name (dico.modules.events.registration.models.registrations.Registration attribute), 186
full_name (dico.modules.rb.models.rooms.Room attribute), 259
full_name (dico.modules.users.models.users.PersonMixin attribute), 242
full_title (dico.modules.events.models.reviews.ProposalGroupProxy attribute), 114
full_title (dico.modules.events.sessions.models.blocks.SessionBlock attribute), 213
full_title (dico.modules.events.tracks.models.tracks.Track attribute), 230
full_title_attr (dico.modules.events.models.reviews.ProposalGroupProxy attribute), 114
full_title_with_group (dico.modules.events.tracks.models.tracks.Track attribute), 230

generate_content ()
(dico.modules.attachments.preview.MarkdownPreviewer class method), 257
generate_content ()
(dico.modules.attachments.preview.Previewer class method), 258
generate_content ()
(dico.modules.attachments.preview.TextPreviewer class method), 258
generate_name ()
(dico.modules.rb.models.rooms.Room method), 299
generate_pdf_from_sessions () (dico.modules.events.sessions.util), 216
generate_spreadsheet_from_abstracts ()
(dico.modules.events.abstracts.util), 137
generate_spreadsheet_from_contributions ()
(dico.modules.events.contributions.util), 156

generate_spreadsheet_from_occurrences ()
(dmodule dico.modules.rb.util), 270
generate_spreadsheet_from_registrations ()
(dico.modules.events.registration.util), 200
generate_spreadsheet_from_sessions ()
(dico.modules.events.sessions.util), 223

generate_spreadsheet_from_survey ()
(dico.modules.events.surveys.util), 223

generate_ticket ()
(dico.modules.events.registration.util), 200
generate_ticket_qr_code (dico.core.signals.event), 69
generate_ticket_qr_code ()
(dico.modules.events.registration.util), 200
get ()
(dico.modules.categories.settings.CategorySettingsProxy method), 240

generate_access_list ()
(dico.modules.events.contributions.models.subcontributions.SubContribution method), 153

generate_active_payment_plugins ()
(dico.modules.events.payment.util), 183
generate_admin_emails ()
(dico.modules.users.util), 249
generate_agreement_definitions ()
(dico.modules.events.agreements.util), 144

generate_all ()
(dico.modules.categories.settings.CategorySettingsProxy method), 241

generate_all ()
(dico.modules.events.settings.EventACLProxy method), 122, 180

generate_all ()
(dico.modules.events.sessions.EventSettingsProxy method), 123, 180

generate_all ()
(dico.modules.categories.settings.CategorySettingsProxy method), 274

generate_all_for_event ()
(dico.modules.events.registration.models.registrations.Registration method), 187

generate_all_templates ()
(dico.modules.designer.util), 283

generate_all_for_event ()
(dico.modules.events.models.events.Event method), 105

generate_all_for_event ()
(dico.modules.events.models.events.Event method), 105

generate_all_for_event ()
(dico.modules.events.models.events.Event method), 105
get_message() (in dico.modules.rb.models.reservations.RepeatMapping class method), 161
get_method_name() (in dico.modules.events.payment.plugins.PaymentPluginMixin method), 184
get_named_default_group() (in dico.modules.groups.core.GroupProxy class method), 276
get_nested_attached_items() (in module dico.modules.attachments.util), 257
get_nested_entries() (in module dico.modules.events.timetable.util), 257
get_next_position() (in module dico.modules.events.tracks.models.tracks), 231
get_non_inheriting_objects() (in dico.modules.events.contributions.models.contributions.Contribution method), 146
get_non_inheriting_objects() (in dico.modules.events.models.events.Event method), 105
get_non_inheriting_objects() (in dico.modules.events.sessions.models.sessions.Session placeholder_list_columns() (in dico.modules.events.registration.settings.RegistrationSettingsProxy method), 204
get_not_deletable_templates() (in module dico.modules.designer.util), 284
get_notification_bcc_list() (in dico.modules.vc.plugins.VCPluginMixin method), 280
get_notification_cc_list() (in dico.modules.vc.plugins.VCPluginMixin method), 280
get_notification_reply_email() (in dico.modules.events.requests.base.RequestDefinitionBase class method), 209
get_notification_template() (in dico.modules.events.requests.base.RequestDefinitionBase class method), 209
get_object_from_args() (in module dico.modules.events.util), 120
get_or_create() (in dico.modules.events.notes.models.notes.EventNote class method), 166
get_or_create_default() (in dico.modules.events.sessions.models.sessions.Session class method), 254
get_overlap()
happens_between() (in module indico.modules.events.models.events.Event method), 106
has_attribute() (in module indico.modules.rb.models.rooms.Room method), 259
has_contributions_with_user_as_submitter() (in module indico.modules.events.contributions.util), 156
has_contributions_with_user_paper_submission_rights() (in module indico.modules.events.papers.util), 179
has_effective_icon (in module indico.modules.events.papers.util), 179
has_ended (in module indico.modules.events.abstracts.models.call_for_abstracts.CallForAbstracts attribute), 128
has_ended (in module indico.modules.events.registration.models.forms.RegistrationForm attribute), 193
has_ended (in module indico.modules.events.papers.models.call_for_papers.CallForPapers attribute), 168
has_ended (in module indico.modules.events.registration.models.forms.RegistrationForm attribute), 193
has_ended (in module indico.modules.events.surveys.models.surveys.Survey attribute), 217
has_equipment() (in module indico.modules.rb.models.rooms.Room method), 260
has_feature() (in module indico.modules.events.models.events.Event method), 106
has_files (in module indico.modules.events.registration.models.registrations.Registration attribute), 187
has_icon (in module indico.modules.categories.models.categories.Category attribute), 235
has_logo (in module indico.modules.categories.models.categories.Category attribute), 235
has_member() (in module indico.modules.groups.core.GroupProxy method), 277
has_note (in module indico.modules.events.sessions.models.blocks.SessionBlock attribute), 235
has_only_events (in module indico.modules.categories.models.categories.Category attribute), 235
has_photo (in module indico.modules.rb.models.rooms.Room attribute), 260
has_role() (in module indico.modules.events.models.persons.EventPerson method), 109
has_sessions_for_user() (in module indico.modules.events.sessions.util), 216
has_started (in module indico.modules.events.abstracts.models.call_for_abstracts.CallForAbstracts attribute), 128
has_started (in module indico.modules.events.papers.models.call_for_papers.CallForPapers attribute), 168
has_started (in module indico.modules.events.registration.models.forms.RegistrationForm attribute), 193
has_started (in module indico.modules.events.surveys.models.surveys.Survey attribute), 217
has_stylesheet (in module indico.modules.events.models.events.Event attribute), 106
has_user_reviewed() (in module indico.modules.events.papers.models.revisions.PaperRevision attribute), 176
has_user_tracks() (in module indico.modules.events.abstracts.models.call_for_abstracts.CallForAbstracts attribute), 138
height_cm (in module indico.modules.designer.pdf.TplData attribute), 284
height (in module indico.modules.designer.pdf.TplData attribute), 284
hidden (indico.modules.networks.models.networks.IPNetworkGroup attribute), 49
HiddenEnumField (class in module indico.forms.fields), 303
HiddenFieldList (class in module indico.forms.fields), 298
html (in module indico.modules.events.layout.models.menu.EventPage attribute), 158
html (in module indico.modules.events.notes.models.notes.EntryNote attribute), 166
html (in module indico.modules.events.notes.models.notes.EntryNoteRevision attribute), 197
html_field_name (in module indico.modules.events.registration.models.form_fields.RegistrationField attribute), 191
html_field_name (in module indico.modules.events.registration.models.form_fields.RegistrationField attribute), 192
id (in module indico.modules.attachments.models.attachments.Attachment attribute), 251
icon (in module indico.modules.categories.models.categories.Category attribute), 235
icon_metadata (in module indico.modules.categories.models.categories.Category attribute), 235
icon_url (in module indico.modules.categories.models.categories.Category attribute), 235
icon_url (in module indico.modules.vc.plugins.VCPluginMixin attribute), 281
id (in module indico.modules.attachments.models.attachments.Attachment attribute), 251
index

368
Index 371
is_paper_reviewer() (in dico.modules.events.contributions.models.contributions.Contribution, 197)
is_required (in dico.modules.events.registration.models.items.RegistrationFormField, 146)
is_parallel () (in dico.modules.events.timetable.models.entries.TimetableEntry, 198)
is_required (in dico.modules.events.registration.models.items.RegistrationFormField, 199)
is_participation (in dico.modules.events.registration.models.forms.RegistrationForm, 226)
is_required (in dico.modules.events.registration.models.items.RegistrationFormField, 193)
is_plugin_link (in dico.modules.events.layout.models.menu.MenuEntryMixin, 243)
is_required (in dico.modules.events.registration.models.items.RegistrationFormField, 219)
is_poster (in dico.modules.events.sessions.models.sessions.Session, 267)
is_required (in dico.modules.events.surveys.models.items.SurveySection, 220)
is_previewable (in dico.modules.attachments.models.attachments.AttachmentFile, 252)
is_required (in dico.modules.events.surveys.models.items.SurveyText, 221)
is_primary (in dico.modules.events.users.models.emails.UserEmail, 246)
is_required (in dico.modules.events.surveys.models.items.SurveySection, 220)
is_private (in dico.modules.events.contributions.models.contributions.Contribution, 154)
is_required (in dico.modules.events.surveys.models.items.SurveySection, 220)
is_protected (in dico.modules.events.contributions.models.contributions.Contribution, 153)
is_required (in dico.modules.events.abstracts.models.abstracts.abstract_submitter, 220)
is_public (in dico.modules.events.contributions.models.contributions.Contribution, 149)
is_required (in dico.modules.events.contributions.models.contributions.Contribution, 146)
is_publishable (in dico.modules.events.registration.models.registrations.Registration, 197)
is_required (in dico.modules.events.registration.models.forms.RegistrationForm, 193)
is_rejected (in dico.modules.rb.models.reservations.Reservation, 269)
is_required (in dico.modules.events.registration.models.items.RegistrationItem, 197)
is_relative (in dico.modules.events.surveys.models.items.SurveySection, 206)
is_required (in dico.modules.events.surveys.models.items.SurveySection, 206)
is_repeating (in dico.modules.rb.models.reservations.Reservation, 206)
is_required (in dico.modules.events.surveys.models.items.SurveySection, 206)
is_request_manager () (in module in dico.modules.events.requests.util, 208)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
is_required (in dico.modules.events.contributions.models.persons.ContributionPersonLink, 277)
is_required (in dico.modules.events.contributions.models.persons.SubContributionPersonLink, 133)
judges (indico.modules.events.abstracts.models.reviews.AbstractCommentVisibility, attribute), 135
judges (indico.modules.events.papers.models.call_for_papers.CallForPapers, attribute), 168
judges (indico.modules.events.papers.models.reviews.PaperCommentVisibility, attribute), 174
judgment_comment (indico.modules.events.abstracts.models.abstracts.Abstract, attribute), 126
judgment_comment (indico.modules.events.papers.models.papers.Paper, attribute), 171
judgment_comment (indico.modules.events.papers.models.revisions.PaperRevision, attribute), 176
judgment_dt (indico.modules.events.abstracts.models.abstracts.Abstract, attribute), 126
judgment_dt (indico.modules.events.papers.models.revisions.PaperRevision, attribute), 176
judgment_instructions (indico.modules.events.abstracts.models.call_for_abstracts.CallForAbstracts, attribute), 128
JudgmentCommentPlaceholder (class in indico.modules.events.abstracts.placeholders, 141
K
key (indico.modules.oauth.models.tokens.OAuthGrant, attribute), 274
key_location (indico.modules.rb.models.rooms.Room, attribute), 260
keywords (indico.modules.events.contributions.models.contributions.Contribution, attribute), 147
keywords (indico.modules.events.models.events.Event, attribute), 106
kind (indico.modules.events.logs.models.entries.EventLogEntry, attribute), 162
L
last_f (indico.modules.users.models.users.NameFormat, attribute), 241
last_f_upper (indico.modules.users.models.users.NameFormat, attribute), 241
last_first (indico.modules.users.models.users.NameFormat, attribute), 241
last_first_upper (indico.modules.users.models.users.NameFormat, attribute), 241
last_login_dt (indico.modules.auth.identities.Identity, attribute), 241
last_login_ip (indico.modules.auth.identities.Identity, attribute), 271
Index 381
Index 383

class method), 118
load() (indico.modules.users.ext.ExtraUserPreferences method), 250
dico.modules.rb.models.blocking_principals.BlockingPrincipal attribute), 263
load_client() (in module in-
dico.modules.oauth.provider), 275
LOCAL_GROUPS (built-in variable), 45
load_grant() (in module in-
dico.modules.oauth.provider), 275
LOCAL_IDENTITIES (built-in variable), 45
load_identity_info() (in module in-
dico.modules.auth.util), 272
LOCAL_OIDENTITIES (built-in variable), (in-
load_token() (in module in-
dico.modules.oauth.provider), 275
LOCAL_MODERATION (built-in variable), 46
local_group (indico.modules.attachments.models.principals.AttachmentFolderPrincipal attribute), 256
LocalGroup (class in in-
local_group (indico.modules.attachments.models.principals.AttachmentFolderPrincipal attribute), 276
localized_title (in-
local_group (indico.modules.categories.models.principals.CategoryPrincipal attribute), 238
local_group (indico.modules.events.contributions.models.principals.ContributionPrincipal in in-
dico.modules.rb.models.locations), 264
local_group (indico.modules.events.models.principals.EventPrincipal in in-
dico.modules.rb.models.rooms.Room attribute), 260
local_group (indico.modules.events.models.settings.EventSettingPrincipal in in-
dico.modules.events.sessions.models.blocks.SessionBlock attribute), 213
local_group (indico.modules.events.sessions.models.principals.SessionPrincipal in in-
dico.modules.rb.models.locations), 260
local_group_id (indico.modules.attachments.models.principals.AttachmentFolderPrincipal f_name), 255
local_group_id (indico.modules.attachments.models.principals.AttachmentPrincipal attribute), 256
local_group_id (indico.modules.categories.models.principals.CategoryPrincipal id (indico.modules.rb.models.rooms.Room attribute), 260
local_group_id (in-
dico.modules.events.contributions.models.principals.ContributionPrincipal in in-
dico.modules.rb.models.rooms.Room attribute), 267
local_group_id (in-
dico.modules.events.models.principals.EventPrincipal in in-
dico.modules.rb.models.rooms.Room attribute), 260
local_group_id (in-
dico.modules.events.models.settings.EventSettingPrincipal in in-
dico.modules.events.contributions.models.contributions.ContributionPrincipal in in-
dico.modules.events.tracks.models.principals.TrackPrincipal in in-
dico.modules.events.sessions.models.principals.SessionPrincipal in in-
dico.modules.events.subconferences.models.subconferences.SubConference attribute), 153
local_group_id (in-
dico.modules.events.tracks.models.principals.TrackPrincipal in in-
dico.modules.events.sessions.models.blocks.SessionBlock attribute), 213
local_group_id (in-
dico.modules.events.tracks.models.principals.TrackPrincipal in in-
dico.modules.events.sessions.models.blocks.SessionBlock attribute), 213
This page contains a list of modules and models, along with attributes related to Indico, a web-based event management system. The page is part of the Indico Documentation, Release 2.3-dev.
map_url (indico.modules.rb.models.rooms.Room attribute), 260
map_url_template (indico.modules.rb.models.locations.Location attribute), 264
MapArea (class in indico.modules.rb.models.map_areas), 264
mapping (indico.modules.rb.models.reservations.RepeatMapping attribute), 265
mark_as_duplicate (indico.modules.events.abstracts.models.reviews.AbstractAction attribute), 134
MarkdownPreviewer (class in indico.modules.attachments.preview), 257
marshmallow_aliases (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 126
marshmallow_aliases (indico.modules.events.abstracts.models.comments.AbstractComment attribute), 129
marshmallow_aliases (indico.modules.events.abstracts.models.reviews.AbstractReview attribute), 135
max_advance_days (indico.modules.rb.models.rooms.Room attribute), 260
MAX_UPLOAD_FILE_SIZE (built-in variable), 54
MAX_UPLOAD_FILES_TOTAL_SIZE (built-in variable), 54
md5 (indico.modules.attachments.models.attachments.AttachmentFile attribute), 252
md5 (indico.modules.designer.models.images.DesignerImageFile attribute), 282
md5 (indico.modules.events.abstracts.models.files.AbstractFile attribute), 132
md5 (indico.modules.events.layout.models.images.ImageFile attribute), 158
md5 (indico.modules.events.papers.models.files.PaperFile attribute), 170
md5 (indico.modules.events.papers.models.templates.PaperTemplate attribute), 178
md5 (indico.modules.events.registration.models.registrations.RegistrationData attribute), 189
md5 (indico.modules.events.static.models.static.StaticSite attribute), 233
meeting (indico.modules.events.models.events.EventType attribute), 109
members (indico.modules.groups.models.groups.LocalGroup attribute), 276
MEMCACHED_SERVERS (built-in variable), 47
menu_entries_for_event() (in module indico.modules.events.layout.util), 161
MenuEntry (class in indico.modules.events.layout.menu), 160
MenuEntryData (class in indico.modules.events.layout.util), 160
MenuEntryMixin (class in indico.modules.events.layout.models.menu), 159
MenuEntryType (class in indico.modules.events.layout.models.menu), 160
merge (indico.modules.events.abstracts.models.reviews.AbstractAction attribute), 134
merge_person_info() (indico.modules.events.models.persons.EventPerson method), 110
merge_users() (in module indico.modules.events.abstracts.models.abstracts.Abstract attribute), 126
merge_users() (in module indico.modules.events.models.persons.EventPerson method), 110
merge_users() (in module indico.modules.events.papers.models.competences.PaperCompetence method), 170
merge_users() (in module indico.core.signals.users), 74
merged (indico.modules.events.abstracts.models.abstracts.AbstractPublicState attribute), 127
merged (indico.modules.events.abstracts.models.abstracts.AbstractState attribute), 128
merged_into (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 126
merged_into_id (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 126
merged_into_id (indico.modules.users.models.users.User attribute), 244
message (indico.modules.events.reminders.models.reminders.EventReminder attribute), 206
message_complete (indico.modules.events.registration.models.forms.RegistrationForm attribute), 193
message_pending (indico.modules.events.registration.models.forms.RegistrationForm attribute), 193
message_unpaid (indico.modules.events.registration.models.forms.RegistrationForm attribute), 193
multipass_group_name

(multipass_group_name

dico.modules.categories.models.principals.CategoryPrincipal

attribute), 238

(multipass_group_name

dico.modules.events.contributions.models.principals.ContributionPrincipal

attribute), 152

(multipass_group_name

dico.modules.events.models.principals.EventPrincipal

attribute), 112

(multipass_group_name

dico.modules.events.models.settings.EventSettingPrincipal

attribute), 117

(multipass_group_name

dico.modules.events.sessions.models.principals.SessionPrincipal

attribute), 215

(multipass_group_name

dico.modules.events.tracks.models.principals.TrackPrincipal

attribute), 231

(multipass_group_name

dico.modules.rb.models.blocking_principals.BlockingPrincipal

attribute), 264

(multipass_group_name

dico.modules.designer.placeholders.RegistrationPositionPlaceholder

attribute), 288

(multipass_group_name

dico.modules.designer.placeholders.RegistrationPhonePlaceholder

attribute), 286

(multipass_group_name

dico.modules.designer.placeholders.RegistrationLastNamePlaceholder

attribute), 285

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNamePlaceholderD

attribute), 285

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNamePlaceholderC

attribute), 284

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNamePlaceholderB

attribute), 284

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderD

attribute), 284

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderC

attribute), 283

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderB

attribute), 283

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholder

attribute), 282

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFriendlyIDPlaceholder

attribute), 281

(multipass_group_name

dico.modules.designer.placeholders.RegistrationFirstNamePlaceholder

attribute), 281

(multipass_group_name

dico.modules.designer.placeholders.RegistrationEmailPlaceholder

attribute), 280

(multipass_group_name

dico.modules.designer.placeholders.RegistrationCountryPlaceholder

attribute), 280

(multipass_group_name

dico.modules.designer.placeholders.RegistrationAmountPlaceholder

attribute), 279

(multipass_group_name

dico.modules.designer.placeholders.RegistrationAffiliationPlaceholder

attribute), 279

(multipass_group_name

dico.modules.designer.placeholders.RegistrationAddressPlaceholder

attribute), 278

(multipass_group_name

dico.modules.designer.placeholders.EventVenuePlaceholder

attribute), 278

(multipass_group_name

dico.modules.designer.placeholders.EventSpeakersPlaceholder

attribute), 277

(multipass_group_name

dico.modules.designer.placeholders.EventRoomPlaceholder

attribute), 276

(multipass_group_name

dico.modules.designer.placeholders.EventOrgTextPlaceholder

attribute), 275

(multipass_group_name

dico.modules.designer.placeholders.EventDescriptionPlaceholder

attribute), 274

(multipass_group_name

dico.modules.designer.placeholders.EventDatesPlaceholder

attribute), 274

(multipass_group_name

dico.modules.categories.models.settings.CategorySetting

attribute), 288

(multipass_group_name

dico.web.forms.fields.MultiStringField

attribute), 304

(multipass_group_name

dico.web.forms.fields.MultipleItemsField

attribute), 304
Index

390

Indico Documentation, Release 2.3-dev

attribute), 210
dico.modules.events.surveys.models.surveys.Survey
attribute), 217
name (dico.modules.groups.models.groups.LocalGroup
attribute), 276
new_tab (dico.modules.events.layout.models.menu.MenuEntry
attribute), 159
name (dico.modules.networks.models.networks.IPNetwork
attribute), 289
NewsItem (class in dico.modules.news.models.news),
290
name (dico.modules.oauth.models.applications.OAuthApplication
attribute), 274
next () (dico.modules.events.payment.models.transactions.TransactionS
class method), 183
name (dico.modules.rb.models.equipment.EquipmentType
attribute), 264
NO_REPLY_EMAIL (built-in variable), 51
name (dico.modules.rb.models.locations.Location at-
tribute), 264
NO_RESOLUTION_USER_STRATEGY (in-
dico.modules.rb.models.reservation_occurrences.ReservationOcc
attribute), 269
name (dico.modules.rb.models.map_areas.MapArea
attribute), 265
nonbookable_periods (in-
dico.modules.rb.models.rooms.Room attribute), 260
name (dico.modules.rb.models.rooms.Room attribute),
260
NonBookablePeriod (class in in-
dico.modules.rb.models.rooms.Room_nonbookable_periods),
261
name (dico.modules.users.models.affiliations.UserAffiliation
attribute), 246
none (dico.modules.events.absorbs.models.absorbs.EditTrackMode
attribute), 128
name (dico.modules.users.models.settings.UserSetting
attribute), 247
none (dico.modules.events.absorbs.models.absorbs.BoAcorrespondingAuthorT
attribute), 142
name (dico.modules.users.models.users.PersonMixin
attribute), 242
none (dico.modules.events.contributions.models.persons.AuthorType
attribute), 150
name (dico.modules.vc.models.vc_rooms.VCRoom at-
tribute), 278
None (class in in-
dico.modules.rb.models.rooms.Room attribute)
name_options (dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholder
attribute), 285
FullName, NoTitlePlaceholder
name_options (dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderB
attribute), 285
name_options (dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderC
attribute), 285
name_options (dico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderD
attribute), 285
not_allowed (dico.modules.events.registration.models.forms.ModificationMode
attribute), 218
name_options (dico.modules.designer.placeholders.RegistrationFullNamePlaceholder
attribute), 286
name_options (dico.modules.designer.placeholders.RegistrationFullNamePlaceholderB
attribute), 286
name_options (dico.modules.designer.placeholders.RegistrationFullNamePlaceholderC
attribute), 286
name_options (dico.modules.designer.placeholders.RegistrationFullNamePlaceholderD
attribute), 286
NameFormat (class in in-
dico.modules.users.models.users), 241
note_added (in module indico.core.signals.event), 70
note_deleted (in module indico.core.signals.event), 70
negative (dico.modules.events.absorbs.models.absorbs.AbstractReviewingState
attribute), 127
note_id (dico.modules.events.notes.models.notes.EventNoteRevision
attribute), 167
negative (dico.modules.events.logs.models.entries.EventLogKind
attribute), 163
note_modified (in module indico.core.signals.event), 70
network (dico.modules.networks.models.networks.IPNetwork
attribute), 289
notification_before_days (in-
dico.modules.events.surveys.models.surveys.Survey
attribute), 70
networks (dico.modules.networks.models.networks.IPNetworkGroup
attribute), 289
NEVER (dico.modules.rb.models.reservations.RepeatFrequency
attribute), 265
new_submission_emails (in-
dico.modules.rb.models.rooms.Room attribute), 260

parent_id (indico.modules.events.surveys.models.items.SurveyItem.attribute), 222
parent_id (indico.modules.events.surveys.models.items.SurveyQuestion.attribute), 220
parent_id (indico.modules.events.surveys.models.items.SurveySection.attribute), 220
parent_id (indico.modules.events.surveys.models.items.Survey.attribute), 221
parent_id (indico.modules.events.timetable.models.entries.TimetableEntry.attribute), 226
partial_completion (indico.modules.events.surveys.models.surveys.Survey.attribute), 218
participants (indico.modules.events.logs.models.entries.EventLogRealm.attribute), 231
participants (indico.modules.events.surveys.models.submissions.SurveySubmission.attribute), 218
pending_paper_files (indico.modules.events.contributions.models.contributions.Contribution.attribute), 147
password (indico.modules.auth.models.identities.Identity.attribute), 107
password_hash (indico.modules.auth.models.identities.Identity.attribute), 271
payment_dt (indico.modules.events.registration.models.registrations.Registration.attribute), 188
PaymentPluginMixin (class in indico.modules.events.payment.plugins), 184
PaymentTransaction (class in indico.modules.events.payment.models.transactions), 182
PDFPreviewer (class in indico.modules.attachments.preview), 257
pending (indico.modules.events.agreements.models.agreements.Agreement.attribute), 133
pending (indico.modules.events.agreements.models.agreements.Agreement.attribute), 144
pending (indico.modules.events.payment.models.transactions.TransactionStatus.attribute), 183
pending (indico.modules.events.payment.models.transactions.TransactionAction.attribute), 183
pending (indico.modules.events.payment.models.transactions.TransactionStatus.attribute), 143
pending (indico.modules.events.payment.models.transactions.TransactionAction.attribute), 195
pending (indico.modules.events.registration.models.registrations.RegistrationState.attribute), 190
pending (indico.modules.events.registration.models.registrations.RegistrationState.attribute), 208
pending (indico.modules.events.static.models.static.StaticSiteState.attribute), 233
pending (indico.modules.rb.models.blocked_rooms.BlockedRoomState.attribute), 151
pending (indico.modules.rb.models.reservations.ReservationState.attribute), 151
pending_answers (indico.modules.events.surveys.models.submissions.SurveySubmission.attribute), 268
person_link_backref_name (indico.modules.events.contributions.models.persons.SubContributionPersonLink.attribute), 151
person_link_backref_name (indico.modules.events.contributions.models.persons.ContributionPersonLink.attribute), 151
person_link_backref_name (indico.modules.events.contributions.models.persons.ContributionPersonLink.attribute), 151
person_link_backref_name (indico.modules.events.contributions.models.persons.AbstractPersonLink.attribute), 133
person_link_backref_name (indico.modules.events.contributions.models.persons.SubContributionPersonLink.attribute), 151
person_link_backref_name (indico.modules.events.contributions.models.persons.ContributionPersonLink.attribute), 151
person_link_backref_name (indico.modules.events.contributions.models.persons.AbstractPersonLink.attribute), 133
person_link_backref_name (indico.modules.events.contributions.models.persons.SubContributionPersonLink.attribute), 151
attribute), 111
person_links (indico.modules.events.sessions.models.blocks.SessionBlockPersonLinkListField, 199)
person_link_backref_name (in dico.modules.events.models.persons.PersonLinkBase, attribute), 214
dico.modules.events.models.persons.PersonLinkBase.person_name (dico.modules.events.agreements.models.agreements.AgreementPersonLink, attribute), 111
person_link_backref_name (in module in dico.modules.events.sessions.models.persons.SessionBlock, person_updated (in module in dico.modules.events.signals.event), 70)
person_updated (in module in dico.modules.events.sessions.models.persons.SessionBlock, personal_data_type, attribute), 214
dico.modules.events.models.persons.SessionBlock.person_updated (in module in dico.modules.events.signals.event), 70
person_link_cls (in dico.modules.events.abstracts.fields.AbstractPersonLinkListField, 191)
person_link_cls (in dico.modules.events.registration.models.form_fields.RegistrationFormPersonLinkListField, 198)
person_link_cls (in dico.modules.events.fields.EventPersonLinkListField, 199)
person_link_cls (in dico.modules.events.contributions.fields.ContributionPersonLinkListField, 199)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListField, 198)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.EventPersonLinkListFieldBase, 291)
person_link_cls (in dico.modules.events.contributions.fields.SubContributionPersonLinkListFieldBase, 291)
Indico Documentation, Release 2.3-dev
registration (indico.modules.events.payment.models.transactions.PaymentTransaction.attribute), 182
registration (indico.modules.events.registration.models.transactions.RegistrationTransaction.attribute), 71
registration_checkin_updated (in module indico.core.signals.event), 70
registration_created (in module indico.core.signals.event), 70
registration_deleted (in module indico.core.signals.event), 70
registration_form_created (in module indico.core.signals.event), 70
registration_form_deleted (in module indico.core.signals.event), 71
registration_form_id (in module indico.modules.events.registration.models.form_fields.RegistrationFormField), 191
registration_form_id (in module indico.modules.events.registration.models.form_fields.RegistrationFormPersonalDataField), 191
registration_form_id (in module indico.modules.events.registration.models.form_fields.RegistrationFormText), 191
registration_form_id (in module indico.modules.events.registration.models.invitations.RegistrationInvitation), 195
registration_form_id (in module indico.modules.events.registration.models.items.RegistrationFormItem), 197
registration_form_id (in module indico.modules.events.registration.models.items.RegistrationFormPersonalDataSection), 198
registration_form_id (in module indico.modules.events.registration.models.items.RegistrationFormSection), 199
registration_form_id (in module indico.modules.events.registration.models.items.RegistrationFormText), 200
registration_form_id (in module indico.modules.events.registration.models.registrations.Registration), 188
registration_id (in module indico.modules.events.payment.models.transactions.PaymentTransaction), 182
registration_id (in module indico.modules.events.registration.models.invitations.RegistrationInvitation), 195
registration_id (in module indico.modules.events.registration.models.items.RegistrationFormItem), 190
registration_limit (in module indico.modules.events.registration.models.forms.RegistrationForm), 194
registration_personal_data_modified (in module indico.core.signals.event), 71
registration_requested (in module indico.core.signals.users), 74
RegistrationAddressPlaceholder (class in indico.modules.designer.placeholders), 287
RegistrationAffiliationPlaceholder (class in indico.modules.designer.placeholders), 287
RegistrationAmountPlaceholder (class in indico.modules.designer.placeholders), 287
RegistrationCountryPlaceholder (class in indico.modules.designer.placeholders), 287
RegistrationData (class in module indico.modules.events.registration.models.registrations), 189
RegistrationForm (class in module indico.modules.events.registration.models.forms), 192
RegistrationFormField (class in module indico.modules.events.registration.models.form_fields), 192
RegistrationFormFieldData (class in module indico.modules.events.registration.models.form_fields), 192
RegistrationFormItem (class in module indico.modules.events.registration.models.items), 196
RegistrationFormItemType (class in module indico.modules.events.registration.models.items), 197
RegistrationFormPersonalDataField (class in module indico.modules.events.registration.models.items), 198
RegistrationFormSection (class in module indico.modules.events.registration.models.items), 199
RegistrationFormText (class in module indico.modules.events.registration.models.items), 200
RegistrationFriendlyIDPlaceholder (class in module indico.modules.designer.placeholders), 287
RegistrationFullNameNoTitlePlaceholder (class in module indico.modules.designer.placeholders), 285
attribute), 206
Request (class in indico.modules.events.requests.models.requests), 207
RequestDefinitionBase (class in indico.modules.events.requests.base), 208
requested_dt (indico.modules.events.static.models.static.StaticSiteAttribute), 233
RequestState (class in indico.modules.events.requests.models.requests), 208
require_feature() (in module indico.modules.events.features.util), 157
require_login (in module indico.modules.events.registration.models.forms.RegistrationForm, 194
require_user (indico.modules.events.registration.models.forms.RegistrationForm, 145
required (indico.modules.events.agreements.placeholders.invitations.InvitationLinkPlaceholder, 203
RescheduleMode (class in indico.modules.events.timetable.reschedule), 229
Rescheduler (class in indico.modules.events.timetable.reschedule), 229
Reservation (class in indico.modules.rb.models.reservations), 265
reservation (indico.modules.rb.models.reservation_edit_logs.ReservationEditLog), 268
reservation (indico.modules.rb.models.reservation_occurrences).reset_state() (in module indico.modules.rb.models.reservations), 172
reservation_id (indico.modules.rb.models.reservation_edit_logs.ReservationEditLog), 268
review (indico.modules.events.abstracts.models.review_ratings.AbstractReviewRating), 134
review (indico.modules.events.abstracts.models.review_ratings.PaperReviewRating), 173
ReservationEditLog (class in indico.modules.rb.models.reservation_edit_logs), 268
ReservationLink (class in indico.modules.rb.models.reservations), 267
ReservationOccurrence (class in indico.modules.rb.models.reservation_occurrences), 268
ReservationOccurrenceState (class in indico.modules.rb.models.reservation_occurrences), 128
reservations (indico.modules.events.models.events.EventAttribute), 108
reservations (indico.modules.rb.models.rooms.RoomAttribute), 260
reservations (indico.modules.users.models.users.UserAttribute), 245
reservations_booked_for (indico.modules.users.models.users.UserAttribute), 245
reservations_need_confirmation (indico.modules.rb.models.rooms.RoomAttribute), 260
ReservationState (class in indico.modules.rb.models.reservations), 268
reset() (indico.modules.events.agreements.models.agreements.Agreement, 137
reset_abstract_state() (in module indico.modules.events.abstracts.models.abstracts.operations), 245
reset_approval() (indico.modules.events.abstracts.operations), 274
reset_client_secret() (indico.modules.users.models.users.UserAttribute), 245
reset_paper_state() (indico.modules.events.papers.operations), 178
reset_signing_secret() (indico.modules.users.models.users.UserAttribute), 245
reset_title() (in module indico.modules.events.abstracts.models.abstracts.EditTrackMode, 280
review (indico.modules.events.abstracts.models.review_ratings.AbstractReviewRating), 134
review (indico.modules.events.abstracts.models.review_ratings.PaperReviewRating), 173
review_class (indico.modules.events.abstracts.models.review_ratings.AbstractReviewRating), 134
review_class (indico.modules.events.abstracts.models.review_ratings.PaperReviewRating), 173
review_id (indico.modules.events.abstracts.models.review_ratings.AbstractReviewRating), 134
review_id (indico.modules.events.abstracts.models.review_ratings.PaperReviewRating), 173
reviewed_for (indico.modules.events.abstracts.models.abstracts.EditTrackMode), 128
attribute), 271
save() (indico.modules.oauth.models.tokens.OAuthGrant
method), 274
save() (indico.modules.users.ext.ExtraUserPreferences
method), 250
save_grant() (in module
indico.modules.oauth.provider), 275
save_identity_info() (in module
indico.modules.auth.util), 273
save_submitted_survey_to_session() (in module
indico.modules.surveys.util), 223
save_token() (in module
indico.modules.oauth.provider), 275
schedule (indico.modules.events.abstracts.settings.BOASortField
dico.modules.users.util), 249
schedule() (indico.modules.events.abstracts.models.call_for_abstracts
method), 128
schedule() (indico.modules.events.abstracts.models.call_for_papers
method), 168
schedule_board_number (indico.modules.events.abstracts.settings.BOASortField
attribute), 142
schedule_cfa() (in module
indico.modules.events.abstracts.operations), 137
schedule_cfp() (in module
indico.modules.events.papers.operations), 178
schedule_contribution() (in module
indico.modules.events.timetable.operations), 227
scheduled_dt (indico.modules.events.reminders.models.reminders.EventReminder
attribute), 206
scheduled_notes (indico.modules.events.models.events.Event
attribute), 108
SCHEDULED_TASK_OVERRIDE (built-in variable), 47
schema_post_dump (in module
indico.core.signals.plugin), 73
schema_post_load (in module
indico.core.signals.plugin), 73
schema_pre_load (in module
indico.core.signals.plugin), 73
scheme (indico.modules.events.models.references.ReferenceType
attribute), 114
scopes (indico.modules.oauth.models.tokens.OAuthToken
attribute), 275
score (indico.modules.events.papers.models.reviews.PaperReview
attribute), 175
score (indico.modules.users.models.suggestions.SuggestedCategory
attribute), 247
search() (indico.modules.groups.core.GroupProxy
class method), 277
search_data (indico.modules.events.registration.models.registrations.
Attribute), 190
search_payload (indico.modules.events.abstracts.fields.AbstractField
attribute), 292
search_url (indico.modules.events.abstracts.fields.AbstractField
attribute), 292
search_users() (in module
indico.core.signals.menu), 72
sections (indico.modules.events.registration.models.items.RegistrationFormItemType
attribute), 197
sections (indico.modules.events.surveys.models.items.SurveyItemType
attribute), 219
sections (indico.modules.events.registration.models.items.RegistrationFormItemType
attribute), 197
sections_with_answered_fields (indico.modules.events.registration.models.registrations.Registration
attribute), 188
send() (indico.modules.events.reminders.models.reminders.EventReminder
method), 207
send() (indico.modules.events.requests.base.RequestDefinitionBase
class method), 210
send_new_agreements() (in module
indico.modules.events.agreements.util), 144
send_start_notification() (indico.modules.events.surveys.models.surveys.Survey
method), 218
sendsubmission_notification() (indico.modules.events.surveys.models.surveys.Survey
method), 218
send_to_participants

session_block_id (in dico.modules.events.sessions.fields, 296)
dico.modules.events.notes.models.notes.EventNoteSessionListToPDF (class in dico.modules.events.sessions.util), 166

session_block_id (in SessionPrincipal (class in dico.modules.events.sessions.models.principals), attribute), 214

session_block_id (in set () (dico.modules.categories.settings.CategorySettingsProxy dico.modules.events.timetable.models.entries.TimetableEntry method), 241)

session_block_id (in set () (dico.modules.events.settings.EventACLProxy method), 123, 180)


attribute), 279

set_attribute_value () (in dico.modules.rb.models.rooms.Room method), 260

module indico.modules.events.sessions.util), 216

set_custom_fields () (in module indico.modules.events.contributions.models.contributions.CustomFieldsMixin method), 148

set_deleted (in module indico.core.signals.event), 71

set_deadline () (in module indico.modules.events.util), 121

session_id (in dico.modules.attachments.models.folders.AttachmentFolder (dico.modules.events.papers.operations), 254

attribute), 178

session_id (in dico.modules.events.contributions.models.contributions.Contribution () (in module dico.modules.events.features.util), 157

session_id (in dico.modules.events.notes.models.notes.EventNoteMulti () (dico.modules.categories.settings.CategorySettingsProxy method), 241)

session_id (in dico.modules.events.sessions.models.blocks.SessionBlock () (dico.modules.categories.settings.CategorySettingsProxy method), 124, 181

session_id (in dico.modules.events.sessions.models.principals.SessionPrincipal (dico.modules.users.models.settings.UserSettingsProxy method), 248

session_id (in dico.modules.rb.models.reservations.ReservationLink: participant_list_columns () (in dico.modules.events.registration.settings.RegistrationSettingsProxy method), 204

SESSION_LIFETIME (built-in variable), 52

set_participant_list_form_ids () (in indico.modules.events.registration.settings.RegistrationSettingsProxy method), 204

attribute), 124, 181

attribute), 241

attribute), 268

set_participant_list_columns () (indico.modules.events.registration.settings.RegistrationSettingsProxy method), 204

attribute), 142

set_participant_list_form_ids () (indico.modules.events.registration.settings.RegistrationSettingsProxy method), 204

attribute), 226

set_reviewing_state () (in module indico.modules.events.timetable.models.entries.TimetableEntry), 178

attribute), 142

session_updated (in module indico.core.signals.event), 71

settings (indico.core.plugins.IndicoPlugin attribute), 64

SessionBlock (class in dico.modules.events.sessions.models.blocks), 212

settings (indico.modules.auth.models.registration_requests.RegistrationRequest attribute), 272

settings (indico.modules.events.settings.ThemeSettingsProxy attribute), 124, 181

settings (indico.modules.users.models.users.User attribute), 245

SessionBlockPersonLinkListField (class in dico.modules.events.sessions.models.persons), 214

settings_backref_name (in-
sprite_position (in-
dico.modules.rb.models.rooms.Room attribute), 260

SQLALCHEMY_DATABASE_URI (built-in variable), 49
SQLALCHEMY_POOL_RECYCLE (built-in variable), 49
SQLALCHEMY_POOL_SIZE (built-in variable), 49
SQLALCHEMY_POOL_TIMEOUT (built-in variable), 49

start_date (indico.modules.rb.models.blockings.Blocking attribute), 262

start_dt (indico.modules.events.abstracts.models.call_for_abstracts.models.Abstract attribute), 128

start_dt (indico.modules.events.contributions.models.contributions.models.Contribution attribute), 148

start_dt (indico.modules.events.models.events.models.Event attribute), 108

start_dt (indico.modules.events.papers.models.papers.models.Paper attribute), 168

start_dt (indico.modules.events.papers.models.call_for_papers.models.CallForAbstracts attribute), 172

start_dt (indico.modules.events.registration.models.forms.models.RegistrationForm attribute), 194

start_dt (indico.modules.events.sessions.models.blocks.models.SessionBlocks attribute), 196

start_dt (indico.modules.events.sessions.models.sessions.models.Session attribute), 188

start_dt (indico.modules.events.surveys.models.surveys.models.Survey attribute), 218

start_notification_sent (in-
dico.modules.events.surveys.models.surveys.models.Survey attribute), 218

start_time (indico.modules.rb.models.room_bookable_hours.models.BookableHours attribute), 261

starts_between () (in-
dico.modules.events.models.events.models.Event attribute), 127

dico.modules.events.models.events.models.Event

state (indico.modules.events.abstracts.models.abstracts.models(Abstract attribute), 148

state (indico.modules.events.agreements.models.agreements.models.Agreement attribute), 144

state (indico.modules.events.papers.models.papers.models.Paper attribute), 162

state (indico.modules.events.papers.models.revisions.models.Revision attribute), 176

state (indico.modules.events.registration.models.invitations.models.Invitation attribute), 184

start_dt (indico.modules.rb.models.reservation_occurrences.models.ReservationOccurrence attribute), 269

start_dt (indico.modules.rb.models.reservations.models.Reservation attribute), 267

start_dt (indico.modules.rb.models.room_nonbookable_periods.models.NonBookablePeriod attribute), 262

start_dt (indico.modules.rb.models.room_bookable_hours.models.BookableHours attribute), 263

start_dt (indico.modules.rb.models.blocked_rooms.modelsBlockedRooms.models.BlockedRoom attribute), 263

start_dt (indico.modules.rb.models.blocked_rooms.modelsBlockedRooms.models.BlockedRoom attribute), 263

start_dt (indico.modules.rb.models.room_bookable_hours.models.BookableHours attribute), 263

start_dt (indico.modules.events.sessions.models.blocks.models.SessionBlocks attribute), 218

STATIC_FILE_METHOD (built-in variable), 53

static_items (indico.modules.events.util.models.ListGeneratorBase attribute), 120

static_list_links (in-
dico.modules.events.models.events.models.Event attribute), 108

StaticListLink (class in in-
dico.modules.events.models.static.models.static.ListGeneratorBase attribute), 120

start_dt (indico.modules.events.models.events.models.Event attribute), 108

start_dt (indico.modules.events.models.events.models.Event attribute), 148

start_notification_emails (in-
dico.modules.events.surveys.models.surveys.models.Survey attribute), 218

start_notification_recipients (in-
dico.modules.events.surveys.models.surveys.models.Survey attribute), 233
<table>
<thead>
<tr>
<th>Class/Attribute</th>
<th>Module/Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatsBase</td>
<td>dico.modules.events.registration.stats, 205</td>
</tr>
<tr>
<td>status</td>
<td>dico.modules.events.payment.models.transactions.PaymentTransaction, 182</td>
</tr>
<tr>
<td>status</td>
<td>dico.modules.vc.models.vc_rooms.VCRoom, 278</td>
</tr>
<tr>
<td>stop_on_match</td>
<td>dico.modules.events.abstracts.models.email_templates.AbstractEmailTemplate, 131</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.attachments.models.attachments.Attachment, 251</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.designer.models.images.DesignerImageFile, 282</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.events.layout.models.images.ImageFile, 158</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.events.papers.models.files.PaperFile, 171</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.events.papers.models.templates.PaperTemplate, 178</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.events.registration.models.registrations.RegistrationData, 190</td>
</tr>
<tr>
<td>storage_backend</td>
<td>dico.modules.events.static.models.static.StaticSite, 233</td>
</tr>
<tr>
<td>STORAGE_BACKENDS</td>
<td>(built-in variable), 52</td>
</tr>
<tr>
<td>storage_file_id</td>
<td>dico.modules.rb.models.reservations.ReservationLink, 268</td>
</tr>
<tr>
<td>storage_file_id</td>
<td>dico.modules.designer.models.images.DesignerImageFile, 282</td>
</tr>
<tr>
<td>storage_file_id</td>
<td>dico.modules.events.contributions.models.contributions.Contribution, 148</td>
</tr>
<tr>
<td>storage_file_id</td>
<td>dico.modules.events.contributions.models.persons.SubContribution, 151</td>
</tr>
<tr>
<td>storage_file_id</td>
<td>dico.modules.events.contributions.models.references.SubContributionReference, 153</td>
</tr>
<tr>
<td>storage_file_id</td>
<td>dico.modules.events.notes.models.notes.EventNote, 166</td>
</tr>
<tr>
<td>store_configuration()</td>
<td>dico.modules.events.util.ListGeneratorBase, 120</td>
</tr>
<tr>
<td>stored_file_class</td>
<td>dico.modules.attachments.models.attachments.Attachment, 251</td>
</tr>
<tr>
<td>stored_file_fkey</td>
<td>dico.modules.attachments.models.attachments.Attachment, 251</td>
</tr>
<tr>
<td>stored_file_table</td>
<td>dico.modules.attachments.models.attachments.Attachment, 251</td>
</tr>
<tr>
<td>stylesheet</td>
<td>dico.modules.events.models.events.Event, 108</td>
</tr>
<tr>
<td>stylesheet_metadata</td>
<td>dico.modules.events.models.events.Event, 108</td>
</tr>
<tr>
<td>SubContribution</td>
<td>dico.modules.events.contributions.models.contributions.Contribution, 148</td>
</tr>
<tr>
<td>subcontribution</td>
<td>dico.modules.events.contributions.models.persons.SubContribution, 151</td>
</tr>
<tr>
<td>subcontribution_count</td>
<td>dico.modules.events.contributions.models.references.SubContributionReference, 153</td>
</tr>
<tr>
<td>subcontribution_created</td>
<td>dico.modules.events.notes.models.notes.EventNote, 166</td>
</tr>
<tr>
<td>subcontribution_deleted</td>
<td>dico.modules.core.signals.event, 71</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.references.SubContributionReference, 153</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.notes.models.notes.EventNote, 166</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.persons.SubContribution, 151</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.references.SubContributionReference, 153</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.notes.models.notes.EventNote, 166</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.persons.SubContribution, 151</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.references.SubContributionReference, 153</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.notes.models.notes.EventNote, 166</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.persons.SubContribution, 151</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.contributions.models.references.SubContributionReference, 153</td>
</tr>
<tr>
<td>subcontribution_id</td>
<td>dico.modules.events.notes.models.notes.EventNote, 166</td>
</tr>
</tbody>
</table>
414

Indico Documentation, Release 2.3-dev

attribute), 166

subcontribution_id (in- attribute), 188

dico.modules.rb.models.reservations.ReservationLink attribute), 268

subcontribution_links (in- submitted_dt (dico.modules.events.surveys.models.submissions.SurveySubmission attribute), 222
dico.modules.events.models.persons.EventPerson attribute), 110

subcontribution_updated (in module in- submitted_for_tracks (dico.modules.events.abstracts.models.abstracts.Abstract

dico.core.signals.event), 71 attribute), 127

SubContributionPersonLink (class in in-
dico.modules.events.contributions.models.persons), subcontribution (dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 127

SubContributionPersonLinkListField (class in in-
dico.modules.events.contributions.fields), 295

subject (dico.modules.events.abstracts.models.email_logs.AbstractEmailLogEntry attribute), 130

SubmitterFirstNamePlaceholder (class in in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 140

SubmitterLastNamePlaceholder (class in in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 140

SubmitterIdPlaceholder (class in in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 140

success (dico.modules.events.static.models.static_StaticSiteState attribute), 233

SubmissionComment (class in in-
dico.modules.events.contributions.models.contributions.Contribution

attribute), 127

SubmitterTitlePlaceholder (class in in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 148

SubmitterNamePlaceholder (class in in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 148

submission_instructions (in-
dico.modules.events.contributions.models.contributions.Contribution

attribute), 127

SurveySubmission (class in in-
dico.modules.events.surveys.models.surveys.Survey

attribute), 183

Submitted (class in in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 128

Submitted (dico.modules.events.papers.models.revisions.PaperRevision attribute), 177

SubmittedContribType (in-
dico.modules.events.contributions.models.contributions.Contribution

attribute), 127

SubmittedContribType_Id (in-
dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 127

SubmittedDt (dico.modules.events.surveys.models.surveys.Survey

attribute), 176

SubmittedDt (dico.modules.events.abstracts.models.abstracts.Abstract

attribute), 127

SubmittedDt (dico.modules.events.registration.models.registrations.Registration

attribute), 180

SubmittedDt (dico.modules.events.rb.models.rooms.Room

attribute), 51

SupportedCurrency () (in-
dico.modules.events.payment.plugins.PaymentPluginMixin attribute), 184

SupportedCurrency (in-
dico.modules.events.payment.plugins.PaymentPluginMixin method), 184
Survey (class in in-dico.modules.events.surveys.models.surveys), 216

survey_id (indico.modules.events.surveys.models.items.SurveyItem attribute), 219

survey_id (indico.modules.events.surveys.models.items.SurveyQuestion attribute), 220

SurveyIDPlaceholder (class in in-dico.modules.events.surveys.models.items.abstracts.placeholders), 141

TargetAbstractTitlePlaceholder (class in indico.modules.events.surveys.models.items.abstracts.placeholders), 141

TargetSubmitterFirstNamePlaceholder (class in indico.modules.events.surveys.models.items.abstracts.placeholders), 141

TargetSubmitterLastNamePlaceholder (class in indico.modules.events.surveys.models.items.abstracts.placeholders), 141

telephone (indico.modules.rb.models.rooms.Room attribute), 261

TEMP_DIR (built-in variable), 50

TEMPATE (indico.modules.attachments.preview.Previewer attribute), 257

TEMPLATE (indico.modules.attachments.preview.ImagePreviewer attribute), 257
template (indico.modules.designer.models.images.DesignerImageFile attribute), 282
template_hook (in module in-dico.core.signals.plugin), 73
template_hook () (indico.core.plugins.IndicoPlugin method), 64
template_id (indico.modules.designer.models.images.DesignerImageFile method), 245
template_kwarg (in module indico.modules.events.logs.renderers.EventLogRendererBase attribute), 164
template_kwarg (in module indico.modules.events.logs.renderers.SimpleRenderer attribute), 164
template_name (in module indico.modules.events.logs.renderers.EmailRenderer attribute), 163
template_name (in module indico.modules.events.logs.renderers.EventLogRendererBase attribute), 164
timezone (indico.web.forms.fields.OccurrencesField attribute), 303
title (indico.modules.events.registration.models.items.RegistrationFormText attribute), 200
timezone_field (indico.web.forms.fields.IndicoDateTimeField attribute), 302
title (indico.modules.events.registration.models.items.RegistrationFormPersonaDataSection attribute), 219
timezone_field (indico.web.forms.fields.OccurrencesField attribute), 303
title (indico.modules.events.registration.models.items.RegistrationFormItem attribute), 255
title (indico.modules.categories.models.categories.Category attribute), 237
title (indico.modules.designer.models.templates.DesignerTemplate attribute), 283
title (indico.modules.events.abstracts.models.abstracts.Abstract attribute), 127
title (indico.modules.events.abstracts.models.email_templates.AbstractEmailTemplate timetable.models.breaks.Break attribute), 225
title (indico.modules.events.abstracts.models.review_questions.AbstractReviewQuestion attribute), 231
title (indico.modules.events.conferences.models.conferences.ConferenceItem attribute), 148
title (indico.modules.events.contributions.models.fields.ContributionField attribute), 149
title (indico.modules.events.contributions.models.subcontributions.models.ContributionBase attribute), 154
title (indico.modules.events.layout.models.menu.MenuEntry title_attr attribute), 159
title (indico.modules.events.models.events.Event attribute), 108
title (indico.modules.events.models.persons.PersonLinkBase to_be_corrected attribute), 111
title (indico.modules.events.models.reviews.ProposalGroupProxy to_be_corrected attribute), 114
title (indico.modules.events.papers.models.reviews.PaperRevisionState title_with_group attribute), 172
title (indico.modules.events.registration.models.form_fields.RegistrationFormField to_dict () attribute), 191
title (indico.modules.events.registration.models.form_fields.RegistrationPersonalDataField to_dict () attribute), 192
title (indico.modules.events.registration.models.form_fields.RegistrationFormField to_dict () attribute), 194
title (indico.modules.events.registration.models.items.PersonalDataType to_dict () attribute), 196
title (indico.modules.events.registration.models.items.RegistrationFormPersonaDataSection to_dict () attribute), 199
top_left_latitude (indico.modules.rb.models.map_areas.MapArea attribute), 200
Track (class in indico.modules.events.tracks.models.tracks), 230
track (indico.modules.events.abstracts.models.reviews.AbstractReview), 238
track (indico.modules.events.contributions.models.contributions.Contribution), 283
track_group (indico.modules.events.tracks.models.tracks.Track), 231
track_id (indico.modules.events.abstracts.models.reviews.AbstractReview), 136
transaction (indico.modules.events.registration.models.registrations.Registration), 188
transaction_id (indico.modules.events.registration.models.registrations.Registration), 188
TransactionStatus (class in indico.modules.events.payment.models.transactions), 183
TransactionStatusTransition (class in indico.modules.events.payment.models.transactions), 183
TransientMenuEntry (class in indico.modules.events.layout.models.menu), 160
translation_domain (indico.core.plugins.IndicoPlugin), 65
time_changes() (in module indico.modules.events.util), 121
TrackPrincipal (class in indico.modules.events.tracks.models.principals), 231
TrackRoleField (class in indico.modules.events.tracks.models.principals), 231
TrackPrincipal (class in indico.modules.events.tracks.models.principals), 231
TrackRoleField (class in indico.modules.events.tracks.models.principals), 231
track_id (indico.modules.events.contributions.models.contributions.Contribution), 148
track_id (indico.modules.events.contributions.models.contributions.Contribution), 152
track_id (indico.modules.events.contributions.models.contributions.Contribution), 162
transaction (indico.modules.events.registration.models.registrations.Registration), 112
transaction_id (indico.modules.events.registration.models.registrations.Registration), 112
TransactionAction (class in indico.modules.events.registration.models.registrations.Registration), 188
TransactionStatus (class in indico.modules.events.payment.models.transactions), 188
TransactionStatusTransition (class in indico.modules.events.payment.models.transactions), 188
TransientMenuEntry (class in indico.modules.events.layout.models.menu), 160
translation_domain (indico.core.plugins.IndicoPlugin), 65
translation_path (indico.core.plugins.IndicoPlugin), 65
ttl (indico.modules.oauth.models.tokens.OAuthGrant), 274
type (indico.modules.events.sessions.models.principals.SessionPrincipal), 215
type (indico.modules.events.sessions.models.sessions.Session), 212
type (indico.modules.events.surveys.models.items.SurveyItem), 191
type (indico.modules.events.surveys.models.items.SurveySection), 175
type (indico.modules.events.surveys.models.items.SurveyText), 191
type (indico.modules.events.surveys.models.items.SurveySection), 191
type (indico.modules.events.surveys.models.items.SurveyText), 191
TrackPrincipal (class in indico.modules.events.tracks.models.principals), 148
TrackRoleField (class in indico.modules.events.tracks.models.principals), 148
TrackPrincipal (class in indico.modules.events.tracks.models.principals), 148
TrackRoleField (class in indico.modules.events.tracks.models.principals), 148
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.events.reviews.models.reviews.Review), 208
item (indico.modules.event...
update_session()

update_session_coordinator_privs()

update_session_block()

update_reviewing_roles()

update_reviewing_question()

update_team_members()

update_review()

update_subcontribution()

update_track_group()

update_timetable_entry()

update_reference_type()

update_session()

update_reviewed_for_tracks()

update_reviewing_question()

update_reviewing_roles()

update_session_block()

update_session_coordinator_privs()

update_state()

update_subcontribution()

update_team_members()

update_timetable_entry()

update_timetable_entry_object()

update_track()

update_track_group()

updated()

url()

url_rule_to_angular()
dico.modules.users.models.affiliations), 245
value (dico.modules.users.models.settings.UserSetting
attribute), 247
vc_room (dico.modules.vc.models.vc_rooms.VCRoomEventAssociation
attribute), 277
vc_room_attach_form (in-
vc_room_form (dico.modules.vc.plugins.VCPluginMixin
attribute), 281
vc_room_id (dico.modules.vc.models.vc_rooms.VCRoomEventAssociation
attribute), 279
VCPluginMixin (class in dico.modules.vc.plugins), 280
Abstract (class in dico.modules.vc.models.vc_rooms), 277
AgreementError, 282
VCRoomEventAssociation (class in in-
dico.modules.vc.models.vc_rooms), 278
VCRoomLinkType (class in in-
dico.modules.vc.models.vc_rooms), 279
VCRoomNotFoundError, 282
cregistry (class in dico.modules.vc.models.vc_rooms), 279
Registration (class in dico.modules.vc.models.vc_rooms), 279
Room (class in dico.modules.rb.models.rooms), 285
verbose_name (in dico.modules.rb.models.rooms), 261
verbose_title (in dico.modules.rb.models.rooms), 261
dico.modules.events.reservation_occurrences.ReservationOccurrenceState
attribute), 269
dico.modules.events.payment.plugins.PaymentPluginMixin
attribute), 184
dico.modules.oauth.models.applications.OAuthApplication
method), 274
dico.modules.categories.models.settings.CategorySetting
attribute), 238
dico.modules.events.abstracts.models.review_ratings.AbstractReviewRating
attribute), 134
dico.modules.events.contributions.models.references.ContributionReference
attribute), 153
dico.modules.events.contributions.models.references.SubContributionReference
attribute), 153
dico.modules.events.models.references.EventReference
attribute), 112
dico.modules.events.models.references.ReferenceModelBase
attribute), 113
dico.modules.events.models.settings.EventSetting
attribute), 116
dico.modules.events.papers.models.review_ratings.PaperReviewRating
attribute), 173
dico.modules.rb.models.room_attributes.RoomAttributeAssociation
attribute), 261
view_data (indico.modules.events.registration.models.itemRegFormSectionField.attribute), 199
view_data (indico.modules.events.registration.models.itemRegFormSectionField.attribute), 200
visibility (indico.modules.categories.models.categories.Category.attribute), 237
visibility (indico.modules.events.abstracts.models.comments.AbstractComment.attribute), 129
visibility (indico.modules.events.abstracts.models.fields.ContributionReviewsFields.attribute), 136
visibility (indico.modules.events.abstracts.models.fields.ContributionReviewsFields.attribute), 149
visibility (indico.modules.events.events.Event.widget attribute), 109
visibility (indico.modules.events.papers.models.comments.PaperReview.attribute), 303
visibility (indico.modules.events.papers.models.reviews.PaperReview.attribute), 175
visibility (indico.modules.events.registration.models.EventPersonLinkListField.attribute), 291
visibility (indico.modules.events.registration.models.items.RegistrationFormSection.attribute), 175
visibility (indico.modules.events.registration.models.items.RegistrationFormSection.attribute), 291
visibility (indico.modules.events.registration.models.items.RegistrationFormSectionBase.attribute), 291
visibility (indico.modules.events.sessions.models.items.SessionBlockPersonLinkListField.attribute), 291
visibility (indico.web.forms.fields.EditableFileField.attribute), 306
visibility (indico.web.forms.fields.IndicoEmailRecipientsField.attribute), 308
visibility (indico.web.forms.fields.IndicoEnumRadioField.attribute), 303
visibility (indico.web.forms.fields.IndicoEnumSelectField.attribute), 303
visibility (indico.web.forms.fields.IndicoLocationField.attribute), 305
was_survey_submitted() (in module indico.modules.events.surveys.util), 223
WEEK (indico.modules.rb.models.reservations.RepeatFrequency.attribute), 265
week_day_data (indico.web.forms.fields.IndicoWeekDayRepetitionField.attribute), 307
WEEK_DAY_NUMBER_CHOICES (indico.web.forms.fields.IndicoWeekDayRepetitionField.attribute), 307
Week (indico.web.forms.fields.IndicoMarkdownField.attribute), 301
Week (indico.web.forms.fields.IndicoNumberField.attribute), 297
Week (indico.web.forms.fields.IndicoRadioField.attribute), 297
Week (indico.web.forms.fields.IndicoSelectMultipleCheckboxField.attribute), 297
Week (indico.web.forms.fields.IndicoStaticTextField.attribute), 301
Week (indico.web.forms.fields.IndicoTagListField.attribute), 301
Week (indico.web.forms.fields.MultiStringField.attribute), 301
Week (indico.web.forms.fields.IndicoStringListField.attribute), 301
Week (indico.web.forms.fields.IndicoTextareaField.attribute), 301
Week (indico.web.forms.fields.IndicoSelectField.attribute), 301
Week (indico.web.forms.fields.IndicoSelectMultipleCheckboxField.attribute), 301
Week (indico.web.forms.fields.IndicoSelectMultipleField.attribute), 301
widget (indico.web.forms.fields.OccurrencesField attribute), 303
widget (indico.web.forms.fields.OverrideMultipleItemsField attribute), 305
widget (indico.web.forms.fields.PrincipalField attribute), 305
widget (indico.web.forms.fields.PrincipalListField attribute), 305
widget (indico.web.forms.fields.RelativeDeltaField attribute), 307
widget (indico.web.forms.fields.TimeDeltaField attribute), 302
width (indico.modules.designer.pdf.TplData attribute), 284
width_cm (indico.modules.designer.pdf.TplData attribute), 284
with_title (indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholder attribute), 285
with_title (indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderB attribute), 285
with_title (indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderC attribute), 286
with_title (indico.modules.designer.placeholders.RegistrationFullNameNoTitlePlaceholderD attribute), 286
with_title (indico.modules.designer.placeholders.RegistrationFullNamePlaceHolder attribute), 285
with_title (indico.modules.designer.placeholders.RegistrationFullNamePlaceHolderB attribute), 285
with_title (indico.modules.designer.placeholders.RegistrationFullNamePlaceHolderC attribute), 285
with_title (indico.modules.designer.placeholders.RegistrationFullNamePlaceHolderD attribute), 286
withdraw() (indico.modules.events.requests.base.RequestDefinitionBase class method), 210
withdraw_abstract() (in module indico.modules.events.abstracts.operations), 137
withdrawn (indico.modules.events.abstracts.models.abstracts.AbstractPublicState attribute), 127
withdrawn (indico.modules.events.abstracts.models.abstracts.AbstractState attribute), 128
withdrawn (indico.modules.events.registration.models.registrations.RegistrationState attribute), 190
withdrawn (indico.modules.events.requests.models.requests.RequestState attribute), 208
WORKER_NAME (built-in variable), 54
WPEventManagement (class in indico.modules.events.management.views), 164

X

XELATEX_PATH (built-in variable), 51