Indico Documentation

Release 1.2

Indico Team

January 08, 2016

Contents

Contents:

CHAPTER 1

indico.util.fossilize – "Serializing" elaborate Python objects to dictionaries and lists

`fossilize` allows us to "serialize" complex python objects into dictionaries and lists. Such operation is very useful for generating JSON data structures from business objects. It works as a wrapper around `zope.interface`.

Some of the features are:

- Different "fossil" types for the same source class;
- Built-in inheritance support;

class indico.util.fossilize.**Fossilizable**
Bases: object

Base class for all the objects that can be fossilized

classmethod clearCache()
Clears the fossil attribute cache

classmethod fossilizeIterable(*target*, *interface*, *useAttrCache=False*, *filterBy=None*, ***kwargs*)
Fossilizes an object, be it a 'direct' fossilizable object, or an iterable (dict, list, set);

classmethod fossilize_obj(*obj*, *interfaceArg=None*, *useAttrCache=False*, *mapClassType={}*, ***kwargs*)
Fossilizes the object, using the fossil provided by *interface*.

Parameters

- **interfaceArg** (*IFossil, NoneType, or dict*) – the target fossile type
- **useAttrCache** (*boolean*) – use caching of attributes if same fields are repeated for a fossil

interface indico.util.fossilize.**IFossil**
Fossil base interface. All fossil classes should derive from this one.

exception indico.util.fossilize.**InvalidFossilException**
Bases: exceptions.Exception

The fossil name doesn't follow the convention I(w+)Fossil or has an invalid method name and did not declare a .name tag for it

exception indico.util.fossilize.**NonFossilizableException**
Bases: exceptions.Exception

Object is not fossilizable (doesn't implement Fossilizable)

indico.util.fossilize.**addFossil**(*klazz*, *fossils*)
Declares fossils for a class

Parameters

- **klazz** – a class object
- **fossils** – a fossil class (or a list of fossil classes)

indico.util.fossilize.**clearCache**()
Shortcut for Fossilizable.clearCache()

indico.util.fossilize.**fossilize**(*target*, *interfaceArg=None*, *useAttrCache=False*, ***kwargs*)
Method that allows the "fossilization" process to be called on data structures (lists, dictionaries and sets) as well as normal *Fossilizable* objects.

Parameters

- **target** (Fossilizable) – target object to be fossilized
- **interfaceArg** (*IFossil, NoneType, or dict*) – target fossil type
- **useAttrCache** (*boolean*) – use the attribute caching

indico.util.fossilize.**fossilizes**(**classList*)
Simple wrapper around 'implements'

1.1 Example

A simple example class:

```
class User(Fossilizable):

    fossilizes(ISimpleUserFossil, IComplexUserFossil)

    def __init__(self, id, name, friends = []):
        self.id = id
        self.name = name
        self.friends = friends

    def getId(self):
        return self.id

    def getName(self):
        return self.name

    def getFriends(self):
        return self.friends
```

(note that the code above will fail if the fossils below are not declared first)

A simple example *Fossil*:

```
class ISimpleUserFossil(IFossil):
    """ A simple user fossil """

    def getId(self):
        """ The ID of the user """

    def getName(self):
```

```
        """ The name, in uppercase """
    getName.convert = lambda x: x.upper()
```

A slightly more complex *Fossil*:

```
class IComplexUserFossil(IFossil):
    """ A complex user fossil """

    def getId(self):
        """ The ID of the user """
    getId.name = 'identityNumber'

    def getFriends(self):
        """ His/her friends """
    getFriends.result = ISimpleUserFossil
```

Output:

```
>>> u1 = User(1,'john')

>>> u1.fossilize(ISimpleUserFossil)
{'id': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'}

>>> u2 = User(2,'bob')

>>> u3 = User(3, 'lisa', friends=[u1,u2])

>>> u3.fossilize(IComplexUserFossil)
{'friends': [{'identityNumber': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'},
             {'id': 2, 'name': 'BOB', '_type': 'User', '_fossil': 'simpleUserFossil'}],
             'id': 3, '_type': 'User', '_fossil': 'complexUserFossil'}

>>> fossilize([u1, u2, u3], ISimpleUserFossil)
[{'id': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'},
 {'id': 2, 'name': 'BOB', '_type': 'User', '_fossil': 'simpleUserFossil'},
 {'id': 3, 'name': 'LISA', '_type': 'User', '_fossil': 'simpleUserFossil'}]
```

1.2 Advanced topics

1.2.1 Valid fossil names. Fossil base class

Valid fossil names have to start with `I` (from "interface") and finish with `Fossil`, i.e. they have to comply with the regular expression:: `^I(\w+)Fossil$` .

Also, fossils have to always inherit directly or indirectly from the `IFossil` fossil, which in turns inherits from `zope.interface.Interface`.

1.2.2 _type and _fossil

All of the fossilized objects produced will have a `_type` attribute, with the name of the original object's class, and a `_fossil` attribute with the name of the fossil used:

```
>>> u = User(1, 'john')
>>> u.fossilize(u, ISimpleUserFossil)
{'id': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'}
```

1.2.3 Valid method names

A fossil's method names have to be in the `get*` form, `has*` form, or `is*` form. Otherwise, the `name` tag is needed. Example:

```
class ISomeFossil(IFossil):
    """ A complex user fossil """

    def getName(self):
        """ The name of the user """

    def hasChildren(self):
        """ Returns if the user has chidlren or not """

    def isMarried(self):
        """ Returns if the user is married or not """

    def requiresAccomodation(self):
        """ Returns if the user requires accomodation or not """
    requiresAccomodation.name = 'requiresAcc'
```

Fossilizing an imaginary user object with this fossil would result in:

```
>>> u.fossilize(ISomeFossil)
{ 'name': 'bob', 'hasChildren': False, 'isMarried': True, 'requiresAcc': True, '_type': 'User', '_fos
```

As shown, the `getXyz` methods correspond to a `xwz` attribute, the `hasXwz` methods correspond to a `xwz` attribute, and so on... The other methods need a `name` tag or an `InvalidFossilException` will be thrown.

1.2.4 Method tags

As seen in the example, it is possible to apply valued tags to the fossil methods:

- `name` tag: overrides the normal name that would be given to the attribute by the fossilizing engine.
- `convert` tag: applies a function to the result of the object's method. Useful to covert datetime objects into strings, capitalize strings, etc.
- `result` tag: when the result of an object's method is another object that might be fossilized, you can specify which interface (fossil) to use with the `result` tag.

1.2.5 Different ways of specifying the fossil to use

Let's take the User class from the first example, and an additional group class. We will not write their methods:

```
class User(Fossilizable):
    """ Class for a User. A User has an id and a name """
    fossilizes(ISimpleUserFossil, IComplexUserFossil)

class Group(Fossilizable):
    """ Class for a Group. A Group has an id and a groupName """
    fossilizes(ISimpleGroupFossil, IComplexGroupFossil)
```

The normal way to specify which fossil to use is to just write the fossil class:

```
>>> u = User(1, 'john')
>>> u.fossilize(u, ISimpleUserFossil)
{'id': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'}
```

This way should be used whenever we are sure that the object we are fossilizing is of a given class.

However, in some cases we are not sure of the interface that should be used. Or, we may be fossilizing a list of heteregenous objects and we cannot or we do not want to use the same fossil for all of them.

In this case, there are currently two options:

- Use `None` as the interface (or leaving the interface argument empty). In this case, the "default" fossil will be used for each object, which means the first fossil declared with the `fossilizes` declaration in the object's class. If the object's class does not invoke `fossilizes` but one of its super-classes does, the first fossil from that super-class will be used. Example:

```
>>> friends = [User(1, 'john'), Group(5, 'family')]
>>> fossilize(friends)
[{'id': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'},
 {'id': 5, 'groupName': 'family', '_type': 'Group', '_fossil': 'simpleGroupFossil'}
```

- Use a dictionary to specify which fossil should be used depending on the object's class. The keys of the dictionary can be: class objects, class names as strings, full class names as strings, or a class object corresponding to an object's super class. Examples:

```
>>> friends = [User(1, 'john'), Group(5, 'family')]
>>> fossilize(friends, {User: ISimpleUserFossil, Group: ISimpleGroupFossil})
[{'id': 1, 'name': 'JOHN', '_type': 'User', '_fossil': 'simpleUserFossil'},
 {'id': 5, 'groupName': 'family', '_type': 'Group', '_fossil': 'simpleGroupFossil'}
>>> fossilize(friends, {"User": ISimpleUserFossil, "Group": ISimpleGroupFossil})
(same output)
>>> fossilize(friends, {"package.subpackage.User": ISimpleUserFossil, "package.subpackage.Group"
(same output)
```

1.2.6 Changing a fossil in execution time

If for some reason you need to change a fossil behaviour in execution time (i.e. after it has been imported), know that it is possible, but **please, avoid doing this unless you have a very good reason for it**. All fossils inherit from zope.interface.Interface, which defines methods so that this is possible.

Example: change the 'name' tag of a given method of a fossil:

```
>>> IComplexUserFossil.get('getFriends').setTaggedValue('name', 'myFriends')
```

CHAPTER 2

indico.tests - test framework

2.1 Architecture

The framework is called through `setup.py` with the argument `test` and optional flags such as `--unit`.

Some dependencies are required:

- `figleaf`
- `nose`
- `selenium`
- `twill`

Each test category is also a class, inheriting from another class named `BaseTestRunner`, which contains useful functions used by all the test classes.

The class `TestManager` is in charge of instantiating each test class and to call their `run()` method.

Some external components such as *JAR* files (for `selenium`/`js-test-driver`) are automatically downloaded. All the information for the downloads is stored in the *tests configuration file*.

2.2 Implementing a new `TestRunner`

To add a new test to the framework, one has to add a new class inheriting from `BaseTestRunner` with a method called `_run()`.

A flag can be added in `setup.py` to run only the new test.

Finally, if for some reason you need to add permanent files in the tests folder, remember to follow the naming convention, which is one folder for each progamming language and one folder for each test category.

CHAPTER 3

indico.modules.scheduler – task scheduling framework and daemon

The scheduler module provides Indico with a scheduling API that allows specific jobs (tasks to be run at given times, with a certain repeatibility, if needed).

3.1 Overview

3.1.1 Architecture

The scheduler module uses the database as the communication medium between "web server processes" (Indico instances, running inside a web server, etc...) and the daemon. This has advantages such as:

- No need for complex IPC mechanisms (RPC, shared memory, etc...);
- Everything is in the Indico DB, which makes migration much easier;

But also poses some problems such as:

- Overhead introduced by regular DB polling on the scheduler (daemon) side;
- Extra database traffic that can slow down things a bit;
- Increased possibility of database conflicts;

We tried to mitigate these problems by using conflict-free lightweight data structures.

The *Scheduler* is the element that is responsible for accepting new tasks and prioritizing them by execution time, launching new processes/threads as they need to be executed. Logs of the operations are kept.

A *Client* is basically a proxy object that allows operations to be performed on the *Scheduler* and its tasks in a transparent way.

3.1.2 Workflow

Tasks can be in one of the following states:

- TASK_STATUS_NONE - Nothing happened yet - this is a transitory state, and normally the state task objects are in when they are created;

- TASK_STATUS_SPOOLED - The task has been added to the spool, and is currently waiting to be put in the waiting queue;
- TASK_STATUS_QUEUED - The algorithm has put the task in the waiting queue;
- TASK_STATUS_RUNNING - The task is being executed;
- TASK_STATUS_FAILED - The task has failed (execution threw an exception, maybe it was cancelled);
- TASK_STATUS_FINISHED - The task has successfully finished;
- TASK_STATUS_TERMINATED - The task has been cancelled by the scheduler (i.e. was AWOL for too long);

...

3.2 Scheduler

The main class in the module is the *Scheduler*

class indico.modules.scheduler.**Scheduler**(***config*)
Bases: object

A *Scheduler* object provides a job scheduler based on a waiting queue, that communicates with its clients through the database. Things have been done in a way that the probability of conflict is minimized, and operations are repeated in case one happens.

The entry point of the process consists of a 'spooler' that periodically takes tasks out of a *conflict-safe* FIFO (spool) and adds them to an IOBTree-based waiting queue. The waiting queue is then checked periodically for the next task, and when the time comes the task is executed.

Tasks are executed in different threads.

The *Client* class works as a transparent remote proxy for this class.

config is a dictionary containing configuration parameters

run()
Main loop, should only be called from scheduler

3.3 Client

Client applications only need to worry about:

class indico.modules.scheduler.**Client**
Bases: object

Client provices a transparent scheduler client, that allows Indico client processes to interact with the Scheduler without the need for a lot of code.

It acts as a remote proxy.

clearSpool()
Clears the spool, returning the number of removed elements

dequeue(*task*)
Schedules a task for deletion

enqueue(*task*)
Schedules a task for execution

getSpool()
Returns the spool

getStatus()

Returns status information (dictionary), containing the lengths (tasks) of:

- spool;
- waiting queue;
- running queue;
- finished task index;
- failed task index;

As well as if the scheduler is running (*state*)

getTask(*tid*)
Returns a *task* object, given its task id

shutdown(*msg=''*)
Shuts down the scheduler. *msg* is an optional paramater that provides an information message that will be written in the logs

startFailedTask(*task*)
Starts a failed task

3.4 Tasks

class indico.modules.scheduler.tasks.**BaseTask**(*expiryDate=None*)
Bases: indico.modules.scheduler.tasks.TimedEvent

A base class for tasks. *expiryDate* is the last point in time when the task can run. A task will refuse to run if current time is past *expiryDate*

prepare()
This information will be saved regardless of the task being repeated or not

reset()
Resets a task to its state before being run

tearDown()
If a task needs to do something once it has run and been removed from runningList, overload this method

class indico.modules.scheduler.tasks.**OneShotTask**(*startDateTime*, *expiryDate=None*)
Bases: *indico.modules.scheduler.tasks.BaseTask*

Tasks that are executed only once

3.5 Module

The module object is of little interest for developers in general. Every Indico instance will transparently provide one through *getDBInstance()*.

class indico.modules.scheduler.**SchedulerModule**
Bases: indico.modules.base.Module

getDBInstance()
Returns the module instance that is stored in the database

getStatus()
Returns some basic info

moveTask(*task*, *moveFrom*, *status*, *occurrence=None*, *nocheck=False*)
Move a task somewhere

removeRunningTask(*task*)
Remove a task from the running list

spool(*op*, *obj*)
Adds an 'instruction' to the spool, in the form (op, obj)

3.6 Example

A simple client use case:

```
>>> from indico.modules.scheduler import Client
>>> from indico.modules.scheduler.tasks import SampleOneShotTask, SamplePeriodicTask
>>> from datetime import timedelta
>>> from dateutil import rrule
>>> from indico.util.date_time import nowutc
>>> c = Client()
>>> st = SampleOneShotTask(nowutc() + timedelta(seconds=1))
>>> c.enqueue(st)
True
>>> dbi.commit()
>>> pt = SamplePeriodicTask(rrule.MINUTELY, bysecond=(40,))
>>> c.enqueue(pt)
True
>>> dbi.commit()
>>> c.dequeue(pt)
>>> dbi.commit()
```

A simple scheduler configuration:

```
s = Scheduler(sleep_interval = 1,
              task_max_tries = 1,
              multitask_mode = 'processes')
```

3.7 Daemon

class indico.modules.scheduler.**Scheduler**(***config*)
Bases: object

A *Scheduler* object provides a job scheduler based on a waiting queue, that communicates with its clients through the database. Things have been done in a way that the probability of conflict is minimized, and operations are repeated in case one happens.

The entry point of the process consists of a 'spooler' that periodically takes tasks out of a *conflict-safe* FIFO (spool) and adds them to an IOBTree-based waiting queue. The waiting queue is then checked periodically for the next task, and when the time comes the task is executed.

Tasks are executed in different threads.

The *Client* class works as a transparent remote proxy for this class.

config is a dictionary containing configuration parameters

run()
Main loop, should only be called from scheduler

CHAPTER 4

Livesync

Livesync is an Indico plugin that allows information to be exported to other systems in a regular basis, and to keep track of what has been exported. It relies on "agents", basically interfaces that convert Indico metadata into some format that can be read by the target system, and negociate the protocol for data delivery.

4.1 The Multipointer Track

The basic data structure used in `livesync` is the *`MultiPointerTrack`* (MPT). It keeps all the information concerning changes that have been done to the data storage, and the curent status of each agent.

class `indico.ext.livesync.struct.`**`MultiPointerTrack`**(*elemContainer*)
Bases: `persistent.Persistent`

A MultiPointerTrack is a kind of structure that is based on an IOBTree, where each entry contains an ordered set (or list, depending on the implementation) of elements. Then, several "pointers" can be created, which point to different positions of the track (very much like runners in a race track). This class is abstract, implementations should be derived.

`add`(*intTS*, *value*)
Adds a value to the container corresponding to a specific timestamp

`addPointer`(*pid*, *startPos=None*)
Registers a new pointer

`getCurrentPosition`(*pid*)
Returns the current entry (set/list) for a given pointer id

`getPointerTimestamp`(*pid*)
Gets the current 'position' of a pointer (id)

`iterate`(*fromPos=None*, *till=None*, *func=<function <lambda>>*)
Generator that iterates through the data structure

`mostRecentTS`(*maximum=None*)
Returns most recent timestamp in track (minimum key) If 'maximum' is provided, return it if less recent

`movePointer`(*pid*, *pos*)
Moves a given pointer (id) to a given timestamp

`oldestTS`()
Returns least recent timestamp in track (maximum key)

pointerIterItems(*pid*, *till=None*)
Iterates over the positions that are left (till the end of the track) for a given pointer (id) - iterates over key-value pairs (iteritems)

pointerIterValues(*pid*, *till=None*)
Iterates over the positions that are left (till the end of the track) for a given pointer (id) - iterates over values

prepareEntry(*ts*)
Creates an empty sub-structure (elemContainer) for a given timestamp

removePointer(*pid*)
Removes a pointer from the list

values(**args*)
Return values or ranges (timestamps) of the structure

class indico.ext.livesync.struct.**SetMultiPointerTrack**
Bases: *indico.ext.livesync.struct.MultiPointerTrack*

OOSet-based MultiPointerTrack implementation. As OOSets are ordered, order is not lost. Order will depend on the __cmp__ method implemented by the contained objects.

4.2 SyncManager

SyncManager is basically a container for agents, and provides an interface for both agent management/querying and basic MPT manipulation.

class indico.ext.livesync.agent.**SyncManager**(*granularity=100*)
Bases: persistent.Persistent

Stores live sync configuration parameters and "agents". It is basically an "Agent Manager"

Parameters **granularity** – integer, number of seconds per MPT entry

add(*timestamp*, *action*)
Adds a specific action to the specified timestamp

advance(*agentId*, *newLastTS*)
Advances the agent "pointer" to the specified timestamp

getAllAgents()
Returns the agent dictionary

classmethod **getDBInstance**()
Returns the instance of SyncManager currently in the DB

getGranularity()
Returns the granularity that is set for the MPT

getTrack()
Rerturns the MPT

objectExcluded(*obj*)
Decides whether a particular object should be ignored or not

query(*agentId=None*, *till=None*)
Queries the agent for a given timespan

registerNewAgent(*agent*)
Registers the agent, placing it in a mapping structure

removeAgent(*agent*)
Removes an agent

reset(*agentsOnly=False*, *trackOnly=False*)
Resets database structures

> **Warning:** This erases any agents and contents in the MPT

4.3 Agents

So far, *PushSyncAgent* is the only available agent type.

class indico.ext.livesync.agent.**SyncAgent**(*aid*, *name*, *description*, *updateTime*, *access=None*)
Bases: *indico.util.fossilize.Fossilizable*, persistent.Persistent

Represents an "agent" (service)

record_str(*(obj*, *objId*, *status)*)
Translates the objects/states to an easy to read textual representation

class indico.ext.livesync.agent.**PushSyncAgent**(*aid*, *name*, *description*, *updateTime*, *access=None*)
Bases: *indico.ext.livesync.agent.SyncAgent*

PushSyncAgents are agents that actively send data to remote services, instead of waiting to be queried.

Parameters

- **aid** – agent ID
- **name** – agent name
- **description** – a description of the agent
- **access** – an Indico user/group that has equivalent access

_generateRecords(*data*, *lastTS*, *dbi=None*)

Parameters

- **data** – iterable containing data to be converted
- **lastTS** –

Takes the raw data (i.e. "event created", etc) and transforms it into a sequence of 'record/action' pairs.

Basically, this function reduces actions to remove server "commands"

i.e. `modified 1234, deleted 1234` becomes just `delete 1234`

Overloaded by agents

_run(*data*, *logger=None*, *monitor=None*, *dbi=None*, *task=None*)
Overloaded - will contain the specific agent code

acknowledge()
Called to signal that the information has been correctly processed (usually called by periodic task)

run(*currentTS*, *logger=None*, *monitor=None*, *dbi=None*, *task=None*)
Main method, called when agent needs to be run

4.4 CLI

There is a Command-line interface available for `livesync`. It can be easily invoked:

```
jdoe $ indico_livesync
usage: indico_livesync [-h] {destroy,list,agent} ...
indico_livesync: error: too few arguments
jdoe $
```

4.4.1 Listing the MPT

It is easy to obtain a listing of what is currently stored in the MPT:

```
jdoe $ indico_livesync list
12970820 <ActionWrapper@0x8df65ac (<MaKaC.conference.Contribution object at 0x8df65ec>) [data_changed
12970819 <ActionWrapper@0x8df662c (<MaKaC.conference.Contribution object at 0x8df65ec>) [data_changed
12970815 <ActionWrapper@0x8df666c (<MaKaC.conference.Contribution object at 0x8df65ec>) [data_changed
12970815 <ActionWrapper@0x8df66ac (<Conference 100994@0x8df67ac>) [data_changed] 1297081528>
12970815 <ActionWrapper@0x8df66ec (<MaKaC.conference.Contribution object at 0x8df65ec>) [data_changed
12970815 <ActionWrapper@0x8df672c (<MaKaC.conference.Category object at 0x8e48dac>) [data_changed] 12
12970815 <ActionWrapper@0x8df676c (<Conference 100994@0x8df67ac>) [data_changed,title_changed,created
```

A query interval can also be specified (`]from, to]`):

```
jdoe $ indico_livesync list --from 12970816 --to 12970819
12970819 <ActionWrapper@0x8db65ac (<MaKaC.conference.Contribution object at 0x8db65ec>) [data_changed
jdoe $
```

4.5 Development Tools

In order to make life easier for plugin developers, we have included a few tools that make things simpler:

class `indico.ext.livesync.agent.`**`RecordUploader`**(*logger*, *agent*, *batchSize=1000*)

Bases: `object`

Encapsulates record uploading behavior.

`_uploadBatch`(*batch*)

Parameters **`batch`** – list of records

To be overloaded by uploaders. Does the actual upload.

`iterateOver`(*iterator*, *dbi=None*)

Consumes an iterator, uploading the records that are returned *dbi* can be passed, so that the cache is cleared once in a while

4.6 Internals

Here are included the listeners and other components that are part of the *livesync* plugin type.

class `indico.ext.livesync.components.`**`ObjectChangeListener`**

Bases: `indico.core.extpoint.base.Component`

This component listens for events and directs them to the MPT. Implements `IAccessControlListener`, `IObjectLifeCycleListener` and `IMetadataChangeListener`

class `indico.ext.livesync.components.`**`RequestListener`**
Bases: `indico.core.extpoint.base.Component`

This component manages the `ContextManager` area that stores livesync actions

class `indico.ext.livesync.components.`**`RequestListenerContext`**
Bases: `object`

This class is mainly useful for testing or CLI scripting

class `indico.ext.livesync.tasks.`**`LiveSyncUpdateTask`**(*frequency*, ***kwargs*)
Bases: `indico.modules.scheduler.tasks.periodic.PeriodicTask`

A task that periodically checks which sources need to be "pushed"

Parameters **`frequency`** – a valid dateutil frequency specifier (DAILY, HOURLY, etc...)

CHAPTER 5

Indico's HTTP Export API

Contents:

5.1 Accessing the API

5.1.1 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 a 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.

5.1.2 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:

```
import hashlib
import hmac
import urllib
import time
```

```
def build_indico_request(path, params, api_key=None, secret_key=None, only_public=False, persistent=F
    items = 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, urllib.urlencode(items))
        signature = hmac.new(secret_key, url, hashlib.sha1).hexdigest()
        items.append(('signature', signature))
    if not items:
        return path
    return '%s?%s' % (path, urllib.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

function build_indico_request($path, $params, $api_key = null, $secret_key = null, $only_public = fal
    if($api_key) {
        $params['apikey'] = $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";
}
```

5.2 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.

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

5.3 API Resources

5.3.1 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

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

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:


```
{
    "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"
            },
            "_type": "Conference",
            "endDate": {
                "date": "2011-06-24",
                "tz": "Europe/Zurich",
                "time": "18:00:00"
            },
            "description": "",
            "title": "Export Test",
            "url": "http://pcituds07.cern.ch/indico/conferenceDisplay.py?confId=137346",
            "location": "CERN",
            "_fossil": "conferenceMetadata",
            "timezone": "Europe/Zurich",
            "type": "meeting",
            "id": "137346",
            "room": null
        }
    ]
}
```


5.3.2 Events

URL Format

/export/event/ID.TYPE

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

Parameters

Param	Short	Description
occurrences	occ	Include the daily event times in the exported data.

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:


```
{
    "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:


```
{
    "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
        },
        {
            "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": "d2c1",
                    "track": null,
                    "duration": 120,
                    "session": null,
                    "location": "CERN",
                    "_fossil": "contributionMetadata",
                    "type": null,
                    "id": "2",
                    "room": null
                }
            ],
            "id": "137346",
            "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:


```
{
    "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": "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": "3",
                            "room": null
                        }
```

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


5.3.3 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:


```
{
    "count": 1,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https:\/\/indico.server\/export\/timetable\/137346.json?ak=00000000-0000-0000-0000-000000
    "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",
                        "tz": "Europe\/Zurich",
                        "time": "16:30:00"
                    },
                    "description": "",
                    "title": "Contrib 1",
                    "id": "c0",
                    "contributionId": "0",
```

```
                    "sessionSlotId": null,
                    "conferenceId": "137346",
                    "presenters": [],
                    "sessionId": null,
                    "location": "CERN",
                    "uniqueId": "a137346t0",
                    "_fossil": "contribSchEntryDisplay",
                    "sessionCode": null,
                    "entryType": "Contribution",
                    "room": "160-1-009"
                }
            }
        }
```


5.3.4 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:


```
{
    "count": 5,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https:\/\/indico.server\/export\/event\/search\/ichep.json?ak=00000000-0000-0000-0000-000
    "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"
        },
        {
            "startDate": {
                "date": "2010-08-06",
                "tz": "UTC",
                "time": "12:00:00"
            },
            "hasAnyProtection": false,
            "id": "102669",
            "title": "Overview of LHC physics results at ICHEP"
```

```
        },
        {
            "startDate": {
                "date": "2010-08-18",
                "tz": "UTC",
                "time": "17:00:00"
            },
            "hasAnyProtection": false,
            "id": "104128",
            "title": "Seminer Oturumu: \"ATLAS status and highlights as of ICHEP\" Dr. Tayfun Ince (U
        },
        {
            "startDate": {
                "date": "2011-07-23",
                "tz": "UTC",
                "time": "11:00:00"
            },
            "hasAnyProtection": false,
            "id": "145521",
            "title": "89th Plenary ECFA and Joint EPS\/ICHEP-ECFA Session - Grenoble, France"
        },
        {
            "startDate": {
                "date": "2012-01-12",
                "tz": "UTC",
                "time": "08:00:00"
            },
            "hasAnyProtection": false,
            "id": "168897",
            "title": "ICHEP 2012 Outreach Planning Meeting"
        }
    ]
}
```


5.3.5 Registration

Registrant list

URL Format

`/export/event/EVENT_ID/registrants.TYPE`

`TYPE` should be `json` or `xml`

Results

Returns the registrant list or error information it there were any problems.

For example:

```
https://indico.server/export/event/0/registrants.json?ak=00000000-0000-0000-0000-000000000000&pretty=
```

Result:

```
{
    "count": 1,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "\/export\/event\/0\/registrants.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes&r
    "ts": 1396431439,
    "results": {
        "registrants": [
            {
                "checkin_secret": "00000000-0000-0000-0000-000000000000",
                "checked_in": true,
                "personal_data": {
                    "city": "Geneva",
                    "fax": "+41227000000",
                    "surname": "Resco Perez",
                    "firstName": "Alberto",
                    "title": "",
                    "country": "CH",
                    "email": "xxxxx.xxxxx.xxxxxx@cern.ch",
                    "phone": "+41227000001",
                    "personalHomepage": "",
                    "address": "",
                    "position": "",
                    "institution": "CERN"
                },
                "full_name": "Alberto Resco Perez",
                "registrant_id": "0"
            }
        ]
    }
}
```


Registrant

URL Format

`/export/event/EVENT_ID/registrant/REGISTRANT_ID.TYPE`

`TYPE` should be `json` or `xml`

Parameters

Param	Values	Description
auth_key	text	Authentication Key in order to be able to get the registrant data

Detail Levels

basic Returns only the personal data of the registrant.

For example:

```
https://indico.server/export/event/0/registrant/0.json?ak=00000000-0000-0000-0000-000000000000&detail
```

Result:


```
{
    "count": 10,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "\/export\/event\/0\/registrant\/0.json?ak=00000000-0000-0000-0000-000000000000&detail=bas
    "ts": 1396431698,
    "results": {
        "_type": "Registrant",
        "checked_in": true,
        "amount_paid": 0,
        "registration_date": "27\/03\/2014 12:20",
        "paid": false,
        "_fossil": "regFormRegistrantBasic",
        "personal_data": {
            "city": "Geneva",
            "fax": "+41227000000",
            "surname": "Resco Perez",
            "firstName": "Alberto",
            "title": "",
            "country": "CH",
            "email": "xxxxx.xxxxx.xxxxxx@cern.ch",
            "phone": "+41227000001",
            "personalHomepage": "",
            "address": "",
            "position": "",
            "institution": "CERN"
        },
        "full_name": "Alberto Resco Perez",
        "checkin_date": "01\/04\/2014 17:27",
        "registrant_id": "0"
    }
}
```


full Returns the full registrant data.

For example:

```
https://indico.server/export/event/0/registrant/0.json?ak=00000000-0000-0000-0000-000000000000&detail
```

Result:


```
{
    "count": 14,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "/export/event/301397/registrant/0.json?ak=00000000-0000-0000-0000-000000000000&detail=ful
    "ts": 1396436802,
    "results": {
        "_type": "Registrant",
        "checked_in": true,
        "amount_paid": 4,
        "registration_date": "24/03/2014 12:42",
        "reasonParticipation": "",
        "paid": true,
        "_fossil": "regFormRegistrantFull",
```

```
        "socialEvents": [],
        "full_name": "Alberto Resco Perez",
        "sessionList": [],
        "checkin_date":  "24/03/2014 12:45",
        "registrant_id": "0",
        "accommodation": {
            "_type": "Accommodation",
            "arrivalDate": "02-04-2014",
            "price": 0,
            "departureDate": "02-04-2014",
            "billable": false,
            "_fossil": "regFormAccommodation",
            "accommodationType": null
        },
        "miscellaneousGroupList": [
            {
                "_fossil": "regFormMiscellaneousInfoGroupFull",
                "_type": "MiscellaneousInfoGroup",
                "id": "0",
                "responseItems": [
                    {
                        "_type": "MiscellaneousInfoSimpleItem",
                        "HTMLName": "*genfield*0-11",
                        "caption": "Personal homepage",
                        "price": 0,
                        "value": "",
                        "currency": "",
                        "_fossil": "regFormMiscellaneousInfoSimpleItem",
                        "id": "11",
                        "quantity": 0
                    },
                    {
                        "_type": "MiscellaneousInfoSimpleItem",
                        "HTMLName": "*genfield*0-10",
                        "caption": "Email",
                        "price": 0,
                        "value": "alberto.resco.perez@cern.ch",
                        "currency": "",
                        "_fossil": "regFormMiscellaneousInfoSimpleItem",
                        "id": "10",
                        "quantity": 0
                    },
                    {
                        "_type": "MiscellaneousInfoSimpleItem",
                        "HTMLName": "*genfield*0-12",
                        "caption": "asdas",
                        "price": "4",
                        "value": 1,
                        "currency": "CHF",
                        "_fossil": "regFormMiscellaneousInfoSimpleItem",
                        "id": "12",
                        "quantity": 1
                    },
                    {
                        "_type": "MiscellaneousInfoSimpleItem",
                        "HTMLName": "*genfield*0-1",
                        "caption": "First Name",
                        "price": 0,
```

```
                        "value": "Alberto",
                        "currency": "",
                        "_fossil": "regFormMiscellaneousInfoSimpleItem",
                        "id": "1",
                        "quantity": 0
                    },
                    ...
                ],
                "title": "Personal Data"
            }
        ]
    }
}
```

Set Paid

URL Format

`/api/event/EVENT_ID/registrant/REGISTRANT_ID/pay.TYPE`

`TYPE` should be `json` or `xml`

Parameters

Param	Values	Description
is_paid	yes, no	If specifed set (or not) as paid

Results

POST request

Returns the status of the payment and the paid amount.

For example:

```
curl --data "ak=00000000-0000-0000-0000-000000000000&is_paid=yes" 'https://indico.server/api/event/0/
```

Result:

```
{
    "count": 2,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "\/api\/event\/301397\/registrant\/0\/pay.json?ak=00000000-0000-0000-0000-000000000000&is_
    "ts": 1396431439,
    "results": {
        "paid": true,
        "amount_paid": 4.0
    }
}
```

Check-in

URL Format

`/api/event/EVENT_ID/registrant/REGISTRANT_ID/checkin.TYPE`

`TYPE` should be `json` or `xml`

Parameters

Param	Values	Description
secret	text	Secret key that gets generated along with the ticket (QR Code)
checked_in	yes, no	If specifed set (or not) as checked in

Results

POST request

Returns the status of the check-in and the check-in date

For example:

```
curl --data "ak=00000000-0000-0000-0000-000000000000&secret=00000000-0000-0000-0000-000000000000&chec
```

Result:

```
{
    "count": 2,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "\/api\/event\/301397\/registrant\/0\/pay.json?ak=00000000-0000-0000-0000-000000000000&sec
    "ts": 1396431439,
    "results": {
        "checked_in": true,
        "checkin_date":  "24/03/2014 12:45",
    }
}
```

5.3.6 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/MATERIA

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

5.3.7 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:


```
{
    "count": 1,
    "additionalInfo": {},
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https:\/\/indico.server\/export\/user\/36024.json?ak=00000000-0000-0000-0000-000000000000
    "ts": 1367243741,
    "results": [
```

```
        {
        "_type": "Avatar",
        "name": "Alberto RESCO PEREZ",
        "firstName": "Alberto",
        "affiliation": "CERN",
        "familyName": "Resco Perez",
        "email": "test@cern.ch",
        "phone": "+41XXXXXXXXX",
        "_fossil": "avatar",
        "title": "",
        "id": "36024"
        }
    ]
}
```


5.3.8 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:

Param	Values	Description
location	text	Room location, e.g. *CERN*
roomid	text	Room id
from/to	f/t	**Start/End time for a booking. Accepted formats:** • 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 it there were any problems.

For example:

```
curl --data "username=jdoe&from=2012-12-30T21:30&to=2012-12-30T22:15&reason=meeting&location=CERN&roo
```

Result:

```
{
    {
        "url": "\/api\/roomBooking\/bookRoom.json",
        "_type": "HTTPAPIResult",
        "results": {
            "reservationID": 45937
        },
        "ts": 1354695663
    }
}
```

Retrieving bookings

General Information The reservation export is only availabled 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.

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

Param	Short	Values	Description
occurrences	occ	yes, no	Include all occurrences of room reservations.
cancelled	cxl	yes, no	If specified only include cancelled (*yes*) or non-cancelled (*no*) reservations.
rejected	rej	yes, no	If specified only include rejected/non-rejected resvs.
confirmed	-	yes, no, pending	If specified only include bookings/pre-bookings with the given state.
archival	arch	yes, no	If specified only include bookings (not) from the past.
recurring	rec	yes, no	If specified only include bookings which are (not) recurring.
repeating	rep	yes, no	Alias for *recurring*
avc	-	yes, no	If specified only include bookings which (do not) use AVC.
avcsupport	avcs	yes, no	If specified only include bookings which (do not) need AVC Support.
startupsup-port	sts	yes, no	If specified only include bookings which (do not) need Startup Support.
bookedfor	bf	text (wildcards)	Only include bookings where the *booked for* field matches the given wildcard string.

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=reservations&from=today&to=today&bookedfor=*MONNICH*&pretty=yes:


```
{
    "count": 1,
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https://indico.server/export/reservation/CERN.json?ak=00000000-0000-0000-0000-0000000000
    "ts": 1308923111,
    "results": [
        {
            "endDT": {
                "date": "2011-06-25",
                "tz": "Europe/Zurich",
                "time": "17:30:00"
            },
            "room": {
                "_fossil": "minimalRoomMetadata",
                "_type": "RoomCERN",
                "fullName": "500-1-201 - Mezzanine",
                "id": 120
            },
            "isConfirmed": true,
            "isValid": true,
            "usesAVC": false,
            "repeatability": "daily",
            "_type": "ReservationCERN",
            "vcList": [],
            "reason": "Just testing",
            "location": "CERN",
            "_fossil": "reservationMetadata",
            "needsAVCSupport": false,
            "startDT": {
                "date": "2011-06-24",
                "tz": "Europe/Zurich",
                "time": "08:30:00"
            },
            "id": 93094,
            "bookingUrl": "http://indico.server/roomBooking.py/bookingDetails?roomLocation=CERN&resv
            "bookedForName": "MONNICH, Jerome"
        }
    ]
}
```


Rooms

General Information

The room export is only availabled 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

Param	Short	Values	Description
occurrences	occ	yes, no	Include all occurrences of room reservations.
cancelled	cxl	yes, no	If specified only include cancelled (*yes*) or non-cancelled (*no*) reservations.
rejected	rej	yes, no	If specified only include rejected/non-rejected resvs.
confirmed	-	yes, no, pending	If specified only include bookings/pre-bookings with the given state.
archival	arch	yes, no	If specified only include bookings (not) from the past.
recurring	rec	yes, no	If specified only include bookings which are (not) recurring.
repeating	rep	yes, no	Alias for *recurring*
avc	-	yes, no	If specified only include bookings which (do not) use AVC.
avcsupport	avcs	yes, no	If specified only include bookings which (do not) need AVC Support.
startupsup-port	sts	yes, no	If specified only include bookings which (do not) need Startup Support.
bookedfor	bf	text (wildcards)	Only include bookings where the *booked for* field matches the given wildcard string.

Detail Levels

rooms Returns basic data about the rooms.

For example, https://indico.server/export/room/CERN/120.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes:


```
{
    "count": 1,
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https://indico.server/export/room/CERN/120.json?ak=00000000-0000-0000-0000-000000000000&p
    "ts": 1308921960,
    "results": [
        {
            "building": 500,
            "_type": "RoomCERN",
            "name": "Mezzanine",
            "floor": "1",
            "longitude": "6.05427049127",
            "vcList": [],
            "equipment": [],
            "roomNr": "201",
            "location": "CERN",
            "_fossil": "roomMetadata",
            "latitude": "46.2314139466",
            "fullName": "500-1-201 - Mezzanine",
            "id": 120,
            "bookingUrl": "http://indico.server/roomBooking.py/bookingForm?roomLocation=CERN&roomID=1
            "avc": false
        }
    ]
}
```

reservations Returns basic data about the rooms and their reservations in the given timeframe.

Output for https://indico.server/export/room/CERN/120.json?ak=00000000-0000-0000-0000-000000000000&detail=reservations&from=today&to=today&pretty=yes:


```
{
    "count": 1,
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https://indico.server/export/room/CERN/120.json?ak=00000000-0000-0000-0000-000000000000&
    "ts": 1308922107,
    "results": [
        {
            "building": 500,
            "_type": "RoomCERN",
            "name": "Mezzanine",
            "floor": "1",
            "longitude": "6.05427049127",
            "reservations": [
                {
                    "endDT": {
                        "date": "2011-06-25",
                        "tz": "Europe/Zurich",
                        "time": "17:30:00"
                    },
                    "isConfirmed": true,
                    "isValid": true,
                    "usesAVC": false,
                    "repeatability": "daily",
                    "_type": "ReservationCERN",
                    "vcList": [],
                    "reason": "Just testing",
                    "bookedForName": "MONNICH, Jerome",
                    "_fossil": "roomReservationMetadata",
                    "needsAVCSupport": false,
                    "startDT": {
                        "date": "2011-06-24",
                        "tz": "Europe/Zurich",
                        "time": "08:30:00"
                    },
                    "id": 93094,
                    "bookingUrl": "http://indico.server/roomBooking.py/bookingDetails?roomLocation=CE
                }
            ],
            "vcList": [],
            "equipment": [],
            "roomNr": "201",
            "location": "CERN",
            "_fossil": "roomMetadataWithReservations",
            "latitude": "46.2314139466",
            "fullName": "500-1-201 - Mezzanine",
            "id": 120,
            "bookingUrl": "http://indico.server/roomBooking.py/bookingForm?roomLocation=CERN&roomID=1
            "avc": false
        }
    ]
}
```

Get room by room name

General Information

The search room export is guest allowed because the room data is public (no the reservations).

URL Format

/export/roomName/LOCATION/ROOMNAME.TYPE

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/Mezzanine.json?ak=00000000-0000-0000-0000-000000000000&pretty=yes:


```
{
    "count": 1,
    "_type": "HTTPAPIResult",
    "complete": true,
    "url": "https://indico.server/export/room/CERN/120.json?ak=00000000-0000-0000-0000-000000000000&p
    "ts": 1308921960,
    "results": [
        {
            "building": 500,
            "_type": "RoomCERN",
            "name": "Mezzanine",
            "floor": "1",
            "longitude": "6.05427049127",
            "vcList": [],
            "equipment": [],
            "roomNr": "201",
            "location": "CERN",
            "_fossil": "roomMetadata",
            "latitude": "46.2314139466",
            "fullName": "500-1-201 - Mezzanine",
            "id": 120,
            "bookingUrl": "http://indico.server/roomBooking.py/bookingForm?roomLocation=CERN&roomID=1
            "avc": false
        }
    ]
}
```

5.3.9 Video Services & Collaboration

URL Format

/export/video/SERVICE_ID.TYPE

The SERVICE_ID may be a single collaboration type or many separated by -. At present, the only TYPE compatible with the Video Services export is *ics* / iCalendar.

As the query is signed with a signature generated using secret API key, the query need not be timestamped. Instead, each booking is given its own unique identifier and, therefore, the generated query URL may be fed as a persistent calendar for importing in your application of choice. The link will only expire once your account has been closed, if TTL is required by your server administrator or your API key is deleted.

If TTL is required by your server administrator, requests should be both timestamped and signed.

Parameters

Param	Short	Values	Description
alarms	-	int	If defined with a value of x int, all bookings to be exported will be accompanied by a matching alarm set to occur x minutes prior to the start of the booking itself. The alarm is set to provide a popup reminder. The default value is 0 minutes.

Please be aware that specifying the alarm parameter in your query will assign alarms to *every* booking which is to be exported.

Service Identifiers Used in CERN

The following parameters are both for example to other installations, and for use within CERN installations of Indico, they represent the options available for configuration through the SERVICE_ID parameter.

SERVICE_ID	Linked Service
all	Traverse all plugin indices.
vidyo	Return Vidyo bookings only.
evo	Return EVO bookings only.
mcu	Return CERNMCU bookings only.
webcast	Return Webcast Requests only.
recording	Return Recording Requests only.

SERVICE_ID may be one of more of these identifiers, if more than one is required simply join the service names with -, please refer to common examples for usage scenarios.

Common Examples

All Bookings

To obtain all bookings in the past 7 days for all collaboration plugins registered:

https://indico.server/export/video/all.ics?ak=API_KEY&from=-70000&to=now&signature=SIGNATURE

To obtain the same output, but with alarms set to display 20 minutes prior to each event:

https://indico.server/export/video/all.ics?ak=API_KEY&alarms=20&from=-70000&to=now&signature=SIGNATURE

Individual Plugin Bookings

To obtain bookings from a specific plugin, in this example Vidyo, from a set date range and with alarms 30 minutes prior to the booking:

https://indico.server/export/video/vidyo.ics?ak=API_KEY&alarms=30&from=2011-08-01&to=2011-12-01&signature=SIGNATURE

Multiple Plugin Bookings

We may also reference more than one plugin's bookings, to request all EVO and CERNMCU bookings over a specified date range with no alarms:

https://indico.server/export/video/evo-mcu.ics?ak=API_KEY&from=2011-09-01&to=2011-09-08&signature=SIGNATURE

5.4 HTTP API Tools

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

i

Symbols

A

B

C

D

E

F

G

I