Contest API 2019

From ICPC-Contest Control Standard
Jump to navigationJump to search

This is the version of the Contest API that was used at WF 2019.

□

Contents

- 1 Introduction
  - 1.1 Changes within this document
  - 1.2 Changes from Contest API 2018
    - 1.2.1 General changes
    - 1.2.2 Endpoint specific changes
- 2 General design principles
  - 2.1 Endpoint URLs
  - 2.2 HTTP headers
  - 2.3 HTTP methods
  - 2.4 Roles
  - 2.5 Referential integrity
  - 2.6 JSON attribute types
  - 2.7 Extensibility
- 3 Interface specification
  - 3.1 Types of endpoints
  - 3.2 Table column description
  - 3.3 Contests
    - 3.3.1 Access restrictions at WF
    - 3.3.2 PATCH start_time
    - 3.3.3 Example
  - 3.4 Judgement Types
    - 3.4.1 Access restrictions at WF
    - 3.4.2 Known judgement types
    - 3.4.3 Examples
  - 3.5 Languages
    - 3.5.1 Access restrictions at WF
    - 3.5.2 Known languages
    - 3.5.3 Example
  - 3.6 Problems
    - 3.6.1 Access restrictions at WF
    - 3.6.2 Examples
  - 3.7 Groups
    - 3.7.1 Access restrictions at WF
    - 3.7.2 Examples

- 3.8 Organizations
  - 3.8.1 Access restrictions at WF
  - 3.8.2 Example
- 3.9 Teams
  - 3.9.1 Access restrictions at WF
  - 3.9.2 Example
- 3.10 Team members
  - 3.10.1 Access restrictions at WF
  - 3.10.2 Example
- 3.11 Contest state
  - 3.11.1 Access restrictions at WF
  - 3.11.2 Example
- 3.12 Submissions
  - 3.12.1 Access restrictions at WF
  - 3.12.2 Example
- 3.13 Judgements
  - 3.13.1 Access restrictions at WF
  - 3.13.2 Example
- 3.14 Runs
  - 3.14.1 Access restrictions at WF
  - 3.14.2 Example
- 3.15 Clarifications
  - 3.15.1 Access restrictions at WF
  - 3.15.2 Examples
- 3.16 Awards
  - 3.16.1 Access restrictions at WF
  - 3.16.2 Semantics
  - 3.16.3 Example
- 3.17 Scoreboard
  - 3.17.1 Scoreboard request options
    - 3.17.1.1 Scoreboard at the time of a given event
  - 3.17.2 Scoreboard format
  - 3.17.3 Access restrictions at WF
  - 3.17.4 Example
- 3.18 Event feed
  - 3.18.1 Feed options
    - 3.18.1.1 Filtering events
    - 3.18.1.2 Feed starting point
  - 3.18.2 Feed format
  - 3.18.3 General access restrictions
  - 3.18.4 Example

Introduction

This page describes an API for accessing information provided by a Contest Control System or Contest Data Server. Such an API can be used by a multitude of clients:

- an external scoreboard
- a scoreboard resolver application

- contest analysis software, such as the ICAT toolset
- another "shadow" CCS, providing forwarding of submissions and all relevant information
- internally, to interface between the CCS server and judging instances

This API is meant to be useful, not only at the ICPC World Finals, but more generally in any ICPC-style contest setup. It is meant to incorporate and supersede the JSON Scoreboard, the REST interface for source code fetching, and the Contest start interface. This REST interface is specified in conjunction with a new NDJSON event feed, which provides all changes to this interface as CRUD-style events and is meant to supersede the old XML Event Feed.

Changes within this document

- Contests
  - The attribute `countdown_pause_time` was changed to be positive instead of negative.

Changes from Contest API 2018

General changes

- The meaning of "should" and "must" were clarified using RFC 2119 and a few occurences of "should" were replaced by "must".
- Implementations must be consistent in the output of the number of decimals in timestamps with sub-second resolution.

Endpoint specific changes

- Contests
  - The attribute `countdown_pause_time` was added.
- State
  - The attribute `started` must be equal to contest `start_time` if set.
  - The attribe `end_of_updates` was added.
  - The order in which state changes can occur and are visible depending on client role were clarified.
- Scoreboard
  - The scoreboard `rows` were moved into a sub-object and attributes `state`, `event`, `time`, and `contest_time` containing metadata were added.
  - An option to request the scoreboard as of a specific event via the parameter `after_event_id` was added.

General design principles

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

The interface is implemented as a HTTP REST interface that outputs information in JSON format (RFC). This REST interface should be provided over HTTPS to guard against eavesdropping on sensitive contest data and authentication credentials (see roles below).

Endpoint URLs

The specific base URL of this API will be dependent on the server (e.g. main CCS or CDS) providing the service; in the specification we only indicate the relative paths of API endpoints with respect to a **baseurl**. In all the examples below the baseurl is `https://example.com/api`.

We follow standard REST practices so that a whole collection can be requested, e.g. at the URL path

```
GET https://example.com/api/contests/wf14/teams
```

while an element with specific ID is requested as

```
GET https://example.com/api/contests/wf14/teams/10
```

A collection is always returned as a JSON list of objects. Every object in the list represents a single element (and always includes the ID). When requesting a single element the exact same object is returned. E.g. the URL path

```
GET baseurl/<collection>
```

returns

```
[ { "id":<id1>, <element specific data for id1>},
  { "id":<id2>, <element specific data for id2>},
    ...
]
```

while the URL path

```
GET baseurl/<collection>/<id1>
```

returns

```
{ "id":<id1>, <element specific data for id1>}
```

HTTP headers

A server should allow cross-origin requests by setting the `Access-Control-Allow-Origin` HTTP header:

```
Access-Control-Allow-Origin: *
```

A server should specify how clients should cache file downloads by setting the `Cache-Control` or `Expires` HTTP headers:

```
Cache-Control: public, max-age=3600, s-maxage=18000
```

```
Expires: Wed, 18 Jul 2018 07:28:00 GMT
```

HTTP methods

The current version of this specification only requires support for the **GET** method, unless explicitly specified otherwise in an endpoint below (see PATCH start_time). However, for future compatibility below are already listed other methods with their expected behavior, if implemented.

GET
: Read data. This method is idempotent and does not modify any data. It can be used to request a whole collection or a specific element.

POST
: Create a new element. This can only be called on a collection endpoint. No **id** attribute should be specified as it is up to the server to assign one, which is returned in the location header.

PUT
: Replaces a specific element. This method is idempotent and can only be called on a specific element and replaces its contents with the data provided. The payload data must be complete, i.e. no partial updates are allowed. The **id** attribute cannot be changed: it does not need to be specified (other than in the URL) and if specified different from in the URL, a **409 Conflict** HTTP code should be returned.

PATCH
: Updates/modifies a specific element. Similar to **PUT** but allows partial updates by providing only that data, for example:
`PATCH https://example.com/api/contests/wf14/teams/10`
with JSON contents
`{"name":"Our cool new team name"}`
No updates of the **id** attribute are allowed either.

DELETE
: Delete a specific element. Idempotent, but may return a 404 status code when repeated. Any provided data is ignored. Example:
`DELETE https://example.com/api/contests/wf14/teams/8`
Note that deletes must keep referential integrity intact.

Standard HTTP status codes are returned to indicate success or failure.

Roles

Access to this API is controlled via user roles. The API provider must require authentication to access each role except for optionally the public role. The API provider must support HTTP basic authentication (RFC). This provides a standard and flexible method; besides HTTP basic auth, other forms of authentication can be offered as well.

Each provider must support at least the following roles, although additional roles may be supported for specific uses:

- public (default role: contest data that's available to everyone)
- admin (data or capability only available to contest administrators)

Role-based access may completely hide some objects from the user, may omit certain attributes, or may embargo or omit objects based on the current contest time. By default, the public user has read-only access (no **POST**, **PUT**, **PATCH** or **DELETE** methods allowed) and does not have access to judgements and runs from submissions made after the contest freeze time.

Referential integrity

Some attributes in elements are references to IDs of other elements. When such an attribute has a non-`null` value, then the referenced element must exist. That is, the full set of data exposed by the API must at all times be referentially intact. This implies for example that before creating a team with an `organization_id`, the organization must already exist. In reverse, that organization can only be deleted after the team is deleted, or alternatively, the team's `organization_id` is set to `null`.

Furthermore, the ID attribute (see below) of elements are not allowed to change. However, note that a particular ID might be reused by first deleting an element and then creating a new element with the same ID.

JSON attribute types

Attribute types are specified as one of the standard JSON types, or one of the more specific types below. Implementations must be consistent with respect to the optional parts of each type, e.g. if the optional .uuu is included in any absolute timestamp it must be included when outputting all absolute timestamps.

Integers
: (type **`integer`** in the specification) are JSON numbers that are restricted to be integer. They should be represented in standard integer representation `(-)?[0-9]+`.

Floating point numbers
: (type **`float`** in the specification) are arbitrary JSON numbers that are expected to take non-integer values. It is recommended to use a decimal representation.

Fixed point numbers
: (type **`decimal`** in the specification) are JSON numbers that are expected to take non-integer values. They must be in decimal (non-scientific) representation and have at most 3 decimals. That is, they must be a integer multiple of `0.001`.

Absolute timestamps
: (type **`TIME`** in the specification) are strings containing human-readable timestamps, given

in ISO 8601 extended combined date/time format with timezone: `yyyy-mm-ddThh:mm:ss(.uuu)?[+-]zz(:mm)?` (or timezone Z for UTC).

Relative times

(type **`RELTIME`** in the specification) are strings containing human-readable time durations, given in a slight modification of the ISO 8601 extended time format: `(-)?(h)*h:mm:ss(.uuu)?`

Identifiers

(type **`ID`** in the specification) are given as string consisting of characters `[a-zA-Z0-9_-]` of length at most 36 and not starting with a `-` (dash). IDs are unique within each endpoint. IDs are assigned by the person or system that is the source of the object, and must be maintained by downstream systems. For example, the person configuring a contest on disk will typically define the ID for each team, and any CCS or CDS that exposes the team must use the same ID.

Some IDs are also used as identifiable labels and are marked below along with the recommended format. These IDs should be meaningful for human communication (e.g. team "43", problem "A") and are as short as reasonable but not more than 10 characters. IDs not marked as labels may be random characters and cannot be assumed to be suitable for display purposes.

Ordinals

(type **`ORDINAL`** in the specification) are used to give an explicit order to a list of objects. Ordinal attributes are integers and must be non-negative and unique in a list of objects, and they should typically be low numbers starting from zero. However, clients must not assume that the ordinals start at zero nor that they are sequential. Instead the ordinal values should be used to sort the list of objects.

File references

(types **`IMAGE`**, **`VIDEO`**, **`ARCHIVE`** and **`STREAM`** in the specification) are represented as a JSON object with elements as defined below.

Element for file reference objects:

Name	Type	Nullable?	Description
href	string	no	URL where the resource can be found. Relative URLs are relative to the **baseurl**. Must point to a file of intended mime-type. Resource must be accessible using the exact same (possibly none) authentication as the call that returned this data.
mime	string	iff default is defined	Mime type of resource. Optional if and only if a default mime-type is specified for the reference.
width	integer	no for **`IMAGE`**	Width of the image, video or stream in pixels. Should not be used for **`ARCHIVE`**.
height	integer	no for **`IMAGE`**	Height of the image, video or stream in pixels. Should not be used for **`ARCHIVE`**.

The **href** attributes may be absolute or relative URLs; relative URLs must be interpreted relative to the **baseurl** of the API. For example, if **baseurl** is `https://example.com/api`, then the following are equivalent JSON response snippets pointing to the same location:

```
"href":"https://example.com/api/contests/wf14/submissions/187/files"
"href":"contests/wf14/submissions/187/files"
```

If implementing support for uploading files pointed to by resource links, substitute the href element with a data element with a base64 encoded string of the associated file contents as the value.

For example

```
POST https://example.com/api/contests/wf14/organizations
```

with JSON data

```
{ "id":"inst105",
  "name":"Carnegie Mellon University",
  ...
  "logo": [{"data": "<base64 string>", "width": 160, "height": 160}]
}
```

Extensibility

This specification is meant to cover the basic data of contests, with the idea that server/client implementations can extend this with more data and/or roles. In particular, this specification already lists some endpoints or specific attributes as optional. The following guidelines are meant to ease extensibility.

- Clients should accept extra attributes in endpoints, that are not specified here.
- Servers should not expect clients to recognize more than the basic, required specification.
- In this specification and extensions, an attribute with value `null` may be left out by the server (i.e. not be present). A client must treat an attribute with value `null` equivalently as that attribute not being present.

Interface specification

The following list of API endpoints should be supported. Note that `state`, `scoreboard` and `event-feed` are singular nouns and indeed contain only a single element.

All endpoints should support **GET**; specific details on other methods are mentioned below.

Types of endpoints

The endpoints can be categorised into 3 groups as follows:

Configuration
: contests, judgement-types, languages, problems, groups, organizations, teams, team-members

Live data
: state, submissions, judgements, runs, clarifications, awards

Aggregate data
: scoreboard, event-feed

Configuration is normally set before contest start. Is not expected to, but could occasionally be updated during a contest. It does not have associated timestamp/contest time attributes. Updates are notified via the event feed.

Live data is generated during the contest and new elements are expected. Data is immutable though, only inserts, no updates or deletes of elements. It does have associated timestamp/contest time attributes. Inserts and deletes are notified via the event feed. **Note**: judgements are the exception to immutability in a weak sense: they get updated once with the final verdict.

Aggregate data: Only **GET** makes sense. These are not included in the event feed, also note that these should not be considered proper REST endpoints, and that the `event-feed` endpoint is a streaming feed in NDJSON format.

Table column description

In the tables below, the columns are:

Name
: Attribute name; object sub-attributes are indicated as `object.attribute`.

Type
: Data type of the attribute; either a JSON type or a type defined above.

Required?
: Whether this is a required attribute that **must** be implemented to conform to this specification.

Nullable?
: Whether the attribute might be `null` (and thus implicitly can also not be present in that case).

Source @WF
: Specifies whether this attribute is implemented at the ICPC World Finals and by whom.

Description
: Description of the meaning of the attribute and any special considerations.

Note that if an attribute is required and nullable, then if in a particular instance it is `null`, it may be left out. On the other hand, an attribute that is optional, but not nullable must either always be present (if the server implements/uses it), or never be present (if the server does not implement/use it). If an attribute is not implemented by a server, then it must never be present; this means that if a client sees an attribute, this means that the server implements it.

Contests

Provides information on the current contest.

The following endpoint is associated with contest:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests	application/json	yes	CDS	JSON array of all contests with elements as defined in the table below
/contests/<id>	application/json	yes	CCS	JSON object of a single contest with elements as defined in the table below

Returns a JSON object with the elements below. If there is no current (this may include about to start or just finished) contest, a 404 error is returned.

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the current contest
name	string	yes	no	CCS	display name of the contest
formal_name	string	no	no	CCS	full name of the contest
start_time	TIME	yes	yes	CCS	the scheduled start time of the contest, may be `null` if the start time is unknown or the countdown is paused
countdown_pause_time	RELTIME	no	yes	CDS	The amount of seconds left when countdown to contest start is paused. At no time may both `start_time` and `countdown_pause_time` be non-`null`.
duration	RELTIME	yes	no	CCS	length of the contest
scoreboard_freeze_duration	RELTIME	no	no	CCS	how long the scoreboard is frozen before the end of the contest
penalty_time	integer	no	no	CCS	penalty time for a wrong submission, in minutes
banner	array of IMAGE	no	yes	CDS	banner for this contest, intended to be an image with a large aspect ratio around 8:1 or so. Default and only allowed mime type is image/png.
logo	array of IMAGE	no	yes	CDS	logo for this contest, intended to be an image with aspect ratio near 1:1. Default and only allowed mime type is image/png.

The expected/typical use of `countdown_pause_time` is that once a `start_time` is defined and close, the countdown may be paused due to unforeseen delays. In this case, `start_time` should be set to `null` and `countdown_pause_time` to the number of seconds left to count down. The `countdown_pause_time` may change to indicate approximate delay. Countdown is resumed by setting a new `start_time` and resetting `countdown_pause_time` to `null`.

Access restrictions at WF

No access restrictions apply to a GET on this endpoint.

PATCH start_time

To replace the Contest Start Interface, at the ICPC World Finals, an API provided by a CCS or CDS implementing this specification must have a role that has the ability to clear or set the contest start time via a PATCH method.

The PATCH must include a valid JSON element with only two attributes allowed: the contest id (used for verification) and a start time (a `<TIME>` value or `null`).

The request should fail with a 401 if the user does not have sufficient access rights, or a 403 if the contest is started or within 30s of starting, or if the new start time is in the past or within 30s.

Example

Request:

```
GET https://example.com/api/contests/wf2014
```

Returned data:

```
{
	"id": "wf2014",
	"name": "2014 ICPC World Finals",
	"formal_name": "38th Annual World Finals of the ACM International Collegiate Programming
Contest",
	"start_time": "2014-06-25T10:00:00+01",
	"duration": "5:00:00",
	"scoreboard_freeze_duration": "1:00:00",
	"penalty_time": 20,
	"banner": [{
		"href": "https://example.com/api/contests/wf2014/banner",
		"width": 1920,
		"height": 240
	}]
}
```

Request:

```
GET https://example.com/api/contests/dress2016
```

Returned data:

```
{
	"id": "dress2016",
	"name": "2016 ICPC World Finals Dress Rehearsal",
	"start_time": null,
	"countdown_pause_time": "0:03:38.749",
	"duration": "2:30:00"
}
```

Request:

```
PATCH https://example.com/api/contests/wf2014
```

Request data:

```
{
	"id": "wf2014",
	"start_time": "2014-06-25T10:00:00+01"
}
```

Request:

```
PATCH https://example.com/api/contests/wf2016
```

Request data:

```
{
        "id": "wf2016",
        "start_time": null
}
```

Judgement Types

Judgement types are the possible responses from the system when judging a submission.

The following endpoints are associated with judgement types:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/judgement-types	application/json	yes	CCS	JSON array of all judgement types with elements as defined in the table below
/contests/<id>/judgement-types/<id>	application/json	yes	CCS	JSON object of a single judgement type with elements as defined in the table below

JSON elements of judgement type objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the judgement type, a 2-3 letter capitalized shorthand, see table below
name	string	yes	no	CCS	name of the judgement. (might not match table below, e.g. if localised)
penalty	boolean	depends	no	CCS	whether this judgement causes penalty time; must be present if and only if contest:penalty_time is present
solved	boolean	yes	no	CCS	whether this judgement is considered correct

Access restrictions at WF

No access restrictions apply to a GET on this endpoint.

Known judgement types

The list below contains standardized identifiers for known judgement types. These identifiers should be used by a server. Please send an email to cliccs@ecs.csus.edu when there are judgement types missing.

The column **Big 5** lists the "big 5" equivalents, if any. A * in the column means that the judgement is one of the "big 5".

The **Translation** column lists other judgements the judgement can safely be translated to, if a system does not support it.

ID	Name	A.k.a.	Big 5	Translation	Description
AC	Accepted	Correct, Yes	*	-	Solves the problem
RE	Rejected	Incorrect, No	WA?	-	Does not solve the problem
WA	Wrong Answer		*	RE	Output is not correct
TLE	Time Limit Exceeded		*	RE	Too slow
RTE	Run-Time Error		*	RE	Crashes
CE	Compile Error		*	RE	Does not compile
APE	Accepted - Presentation Error	Presentation Error, also see AC and PE	AC	AC	Solves the problem, although formatting is wrong
OLE	Output Limit Exceeded		WA	WA, RE	Output is larger than allowed
PE	Presentation Error	Output Format Error	WA	WA, RE	Data in output is correct, but formatting is wrong
EO	Excessive Output		WA	WA, RE	A correct output is produced, but also additional output
IO	Incomplete Output		WA	WA, RE	Parts, but not all, of a correct output is produced
NO	No Output		WA	IO, WA, RE	There is no output
WTL	Wallclock Time Limit Exceeded		TLE	TLE, RE	CPU time limit is not exceeded, but wallclock is
ILE	Idleness Limit Exceeded		TLE	WTL, TLE, RE	No CPU time used for too long
TCO	Time Limit Exceeded - Correct Output		TLE	TLE, RE	Too slow but producing correct output
TWA	Time Limit Exceeded - Wrong Answer		TLE	TLE, RE	Too slow and also incorrect output
TPE	Time Limit Exceeded - Presentation Error		TLE	TWA, TLE, RE	Too slow and also presentation error
TEO	Time Limit Exceeded - Excessive Output		TLE	TWA, TLE, RE	Too slow and also excessive output
TIO	Time Limit Exceeded - Incomplete Output		TLE	TWA, TLE, RE	Too slow and also incomplete output
TNO	Time Limit Exceeded - No Output		TLE	TIO, TWA, TLE, RE	Too slow and also no output
MLE	Memory Limit Exceeded		RTE	RTE, RE	Uses too much memory
SV	Security Violation	Illegal Function, Restricted Function	RTE	RTE, RE	Uses some functionality that is not allowed by the system
IF	Illegal Function	Illegal Function, Restricted Function	RTE	SV, RTE, RE	Calls a function that is not allowed by the system
RCO	Run-Time Error - Correct Output		RTE	RTE, RE	Crashing but producing correct output
RWA	Run-Time Error - Wrong Answer		RTE	RTE, RE	Crashing and also incorrect output
RPE	Run-Time Error - Presentation Error		RTE	RWA, RTE, RE	Crashing and also presentation error
REO	Run-Time Error - Excessive Output		RTE	RWA, RTE, RE	Crashing and also excessive output
RIO	Run-Time Error - Incomplete Output		RTE	RWA, RTE, RE	Crashing and also incomplete output
RNO	Run-Time Error - No Output		RTE	RIO, RWA, RTE, RE	Crashing and also no output
CTL	Compile Time Limit Exceeded		CE	CE, RE	Compilation took too long
JE	Judging Error		-	-	Something went wrong with the system
SE	Submission Error		-	-	Something went wrong with the submission
CS	Contact Staff	Other	-	-	Something went wrong

Examples

Request:

```
GET https://example.com/api/contests/wf14/judgement-types
```

Returned data:

```
[{
        "id": "CE",
        "name": "Compiler Error",
        "penalty": false,
        "solved": false
}, {
        "id": "AC",
        "name": "Accepted",
        "penalty": false,
        "solved": true
}]
```

Request:

```
GET https://example.com/api/contests/wf14/judgement-types/AC
```

Returned data:

```
{
        "id": "AC",
        "name": "Accepted",
        "penalty": false,
        "solved": true
}
```

Languages

Languages that are available for submission at the contest.

The following endpoints are associated with languages:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/languages	application/json	yes	CCS	JSON array of all languages with elements as defined in the table below
/contests/<id>/languages/<id>	application/json	yes	CCS	JSON object of a single language with elements as defined in the table below

JSON elements of language objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the language from table below
name	string	yes	no	CCS	name of the language (might not match table below, e.g. if localised)

Access restrictions at WF

No access restrictions apply to a GET on this endpoint.

Known languages

Below is a list of standardized identifiers for known languages. When providing one of these languages, the corresponding identifier should be used. The language name may be adapted e.g. for localization or to indicate a particular version of the language. In case multiple versions of a language are provided, those must have separate, unique identifiers. It is recommended to choose new identifiers with a suffix appended to an existing one. For example `cpp17` to specify the ISO 2017 version of C++.

ID	Name
ada	Ada
c	C
cpp	C++
csharp	C#
go	Go
haskell	Haskell
java	Java
javascript	JavaScript
kotlin	Kotlin
objectivec	Objective-C
pascal	Pascal
php	PHP
prolog	Prolog
python2	Python 2
python3	Python 3
ruby	Ruby
rust	Rust
scala	Scala

Example

Request:

```
GET https://example.com/api/contests/wf14/languages
```

Returned data:

```
[{
        "id": "java",
        "name": "Java"
}, {
        "id": "cpp",
        "name": "GNU C++"
}, {
        "id": "python2",
```

```
        "name": "Python 2"
}]
```

Problems

The problems to be solved in the contest

The following endpoints are associated with problems:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/problems	application/json	yes	CCS	JSON array of all problems with elements as defined in the table below
/contests/<id>/problems/<id>	application/json	yes	CCS	JSON object of a single problem with elements as defined in the table below

JSON elements of problem objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the problem, at the WFs the directory name of the problem archive
label	string	yes	no	CCS	label of the problem on the scoreboard, typically a single capitalized letter
name	string	yes	no	CCS	name of the problem
ordinal	ORDINAL	yes	no	CCS	ordering of problems on the scoreboard
rgb	string	no	no	CCS	hexadecimal RGB value of problem color as specified in HTML hexadecimal colors, e.g. '#AC00FF' or '#fff'
color	string	no	no	CCS	human readable color description associated to the RGB value
time_limit	decimal	no	no	CCS	time limit in seconds per test data set (i.e. per single run)
test_data_count	integer	yes	no	CCS	number of test data sets

Access restrictions at WF

The **public** role can only access these problems after the contest started. That is, before contest start this endpoint returns an empty array for clients with the **public** role.

Examples

Request:

```
GET https://example.com/api/contests/wf14/problems
```

Returned data:

```
[{"id":"asteroids","label":"A","name":"Asteroid
```

```
Rangers","ordinal":1,"color":"blue","rgb":"#00f","time_limit":2,"test_data_count":10},
  {"id":"bottles","label":"B","name":"Curvy Little
Bottles","ordinal":2,"color":"gray","rgb":"#808080","time_limit":3.5,"test_data_count":15}
]
```

Request:

```
GET https://example.com/api/contests/wf14/problems/asteroids
```

Returned data:

```
{"id":"asteroids","label":"A","name":"Asteroid
Rangers","ordinal":1,"color":"blue","rgb":"#00f","time_limit":2,"test_data_count":10}
```

Groups

Grouping of teams. At the World Finals these are the super regions, at regionals these are often different sites.

The following endpoints are associated with groups:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/groups	application/json	no	CCS	JSON array of all groups with elements as defined in the table below
/contests/<id>/groups/<id>	application/json	no	CCS	JSON object of a single group with elements as defined in the table below

Note that these endpoints must be provided if groups are used. If they are not provided no other endpoint may refer to groups (i.e. return any group_ids).

JSON elements of group objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the group
icpc_id	string	no	yes	CCS	external identifier from ICPC CMS
name	string	yes	no	CCS	name of the group
type	string	no	yes	CCS	type of this group
hidden	boolean	no	yes	CCS	if group should be hidden from scoreboard. Defaults to false if missing.

Access restrictions at WF

No access restrictions apply to a GET on this endpoint.

Examples

Request:

```
GET https://example.com/api/contests/wf14/groups
```

Returned data:

```
[{"id":"asia-74324325532","icpc_id":"7593","name":"Asia"}
]
```

Request:

```
GET https://example.com/api/contests/wf14/groups
```

Returned data:

```
[{"id":"42425","name":"Division 2","type":"division"}
]
```

Organizations

Teams can be associated with organizations which will have some associated information, e.g. a logo. Typically organizations will be universities.

The following endpoints are associated with organizations:

Endpoint	Type	Required?	Source @WF	Description
/contests/<id>/organizations	application/json	no	CCS & CDS	JSON array of all organizations with elements as defined in the table below
/contests/<id>/organizations/<id>	application/json	no	CCS & CDS	JSON object of a single organization with elements as defined in the table below

Note that the first two endpoints must be provided if organizations are used. If they are not provided no other endpoint may refer to organizations (i.e. return any organization_ids).

JSON elements of organization objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the organization
icpc_id	string	no	yes	CCS	external identifier from ICPC CMS
name	string	yes	no	CCS	display name of the organization
formal_name	string	no	yes	CCS	full organization name if too long for normal display purposes.
country	string	no	yes	not used	ISO 3-letter code of the organization's country
url	string	no	yes	CDS	URL to organization's website
twitter_hashtag	string	no	yes	CDS	organization hashtag
location	object	no	yes	CDS	JSON object as specified in the rows below
location.latitude	float	depends	no	CDS	Latitude in degrees. Required iff location is present.
location.longitude	float	depends	no	CDS	Longitude in degrees. Required iff location is present.
logo	array of IMAGE	no	yes	CDS	logo of the organization. Default and only allowed mime type is image/png. A server must provide logos of size 56x56 and 160x160 but may provide other sizes as well.

Access restrictions at WF

No access restrictions apply to a GET on organizations endpoints.

Example

Request:

```
GET https://example.com/api/contests/<id>/organizations
```

Returned data:

```
[{"id":"inst123","icpc_id":"433","name":"Shanghai Jiao Tong U.","formal_name":"Shanghai Jiao Tong
University"},
  {"id":"inst105","name":"Carnegie Mellon University","country":"USA",
"logo":[{"href":"http://example.com/api/contests/wf14/organizations/inst105/logo/56px","width":56,"h
eight":56},
{"href":"http://example.com/api/contests/wf14/organizations/inst105/logo/160px","width":160,"height"
:160}]
  }
 ]
```

Teams

Teams competing in the contest.

The following endpoints are associated with teams:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/teams	application/json	yes	CCS & CDS	JSON array of all teams with elements as defined in the table below
/contests/<id>/teams/<id>	application/json	yes	CCS & CDS	JSON object of a single team with elements as defined in the table below

JSON elements of team objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the team. Usable as a label, at WFs normally the team seat number
icpc_id	string	no	yes	CCS	external identifier from ICPC CMS
name	string	yes	no	CCS	name of the team
organization_id	ID	no	yes	CCS	identifier of the organization (e.g. university or other entity) that this team is affiliated to
group_ids	array of ID	no	no	CCS	identifiers of the group(s) this team is part of (at ICPC WFs these are the super-regions). No meaning must be implied or inferred from the order of IDs. The array may be empty.
location	object	no	no	CDS	JSON object as specified in the rows below
location.x	float	depends	no	CDS	Team's x position in meters. Required iff location is present.
location.y	float	depends	no	CDS	Team's y position in meters. Required iff location is present.
location.rotation	float	depends	no	CDS	Team's rotation in degrees. Required iff location is present.
photo	array of IMAGE	no	yes	CDS	registration photo of the team. Default mime type is image/png
video	array of VIDEO	no	yes	CDS	registration video of the team.
backup	array of ARCHIVE	no	yes	CDS	latest file backup of the team machine. Default and only allowed mime type is application/zip.
desktop	array of STREAM	no	yes	CDS	streaming video of the team desktop.
webcam	array of STREAM	no	yes	CDS	streaming video of the team webcam.

Access restrictions at WF

The following access restrictions apply to a GET on this endpoint:

- **backup** requires the **admin** or **analyst** role for access,
- the **desktop** and **webcam** attributes are available for the **public** role only when scoreboard is not frozen.

Example

Request:

```
GET https://example.com/api/contests/wf14/teams
```

Returned data:

```
[{"id":"11","icpc_id":"201433","name":"Shanghai
Tigers","organization_id":"inst123","group_ids":["asia-74324325532"]},
  {"id":"123","name":"CMU1","organization_id":"inst105","group_ids":["8","11"]}
```

```
]
```

Team members

Team members of teams in the contest.

The following endpoints are associated with languages:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/team-members	application/json	no	CDS	JSON array of all team members with elements as defined in the table below
/contests/<id>/team-members/<id>	application/json	no	CDS	JSON object of a single team member with elements as defined in the table below

JSON elements of team member objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CDS	identifier of the team-member
icpc_id	string	no	yes	CDS	external identifier from ICPC CMS
team_id	ID	yes	no	CDS	team of this team member
first_name	string	yes	no	CDS	first name of team member
last_name	string	yes	no	CDS	last name of team member
sex	string	no	yes	CDS	either **male** or **female**, or possibly `null`
role	string	yes	no	CDS	one of **contestant** or **coach**
photo	array of IMAGE	no	yes	CDS	registration photo of the team member. Default mime type is image/png

Access restrictions at WF

No access restrictions apply to a GET on this endpoint.

Example

Request:

```
GET https://example.com/api/contests/wf14/team-members
```

Returned data:

```
[{"id":"john-
smith","team_id":"43","icpc_id":"32442","first_name":"John","last_name":"Smith","sex":"male","role":
"contestant"},
  {"id":"osten-
umlautsen","team_id":"43","icpc_id":null,"first_name":"Östen","last_name":"Ümlautsen","sex":null,"ro
le":"coach"}
]
```

Contest state

Current state of the contest, specifying whether it's running, the scoreboard is frozen or results are final.

The following endpoints are associated with state:

Endpoint	Type	Required?	Source @WF	Description
/contests/<id>/state	application/json	yes	CCS	JSON object of the current contest state with elements as defined in the table below

JSON elements of state objects:

Name	Type	Required?	Nullable?	Source @WF	Description
started	TIME	yes	yes	CCS	Time when the contest actually started, or `null` if the contest has not started yet. When set, this time must be equal to the contest `start_time`.
frozen	TIME	depends	yes	CCS	Time when the scoreboard was frozen, or `null` if the scoreboard has not been frozen. Required iff `scoreboard_freeze_duration` is present in the contest endpoint.
ended	TIME	yes	yes	CCS	Time when the contest ended, or `null` if the contest has not ended. Must not be set if started is `null`.
thawed	TIME	depends	yes	CCS	Time when the scoreboard was thawed (that is, unfrozen again), or `null` if the scoreboard has not been thawed. Required iff `scoreboard_freeze_duration` is present in the contest endpoint. Must not be set if frozen is `null`.
finalized	TIME	yes	yes	CCS	Time when the results were finalized, or `null` if results have not been finalized. Must not be set if ended is `null`.
end_of_updates	TIME	yes	yes	CCS	Time after last update to the contest occurred, or `null` if more updates are still to come. Setting this to non-`null` must be the very last change in the contest.

These state changes must occur in the order listed in the table above, as far as they do occur, except that `thawed` and `finalized` may occur in any order. For example, the contest may never be frozen and hence not thawed either, or, it may be finalized before it is thawed. That, is the following sequence of inequalities must hold:

```
started < frozen < ended < thawed    < end_of_updates,
                   ended < finalized < end_of_updates.
```

A contest that has ended, has been thawed (or was never frozen) and is finalized must not change. Thus, `end_of_updates` can be set once both `finalized` is set and `thawed` is set if the contest was frozen.

Access restrictions at WF

No access restrictions apply to a GET on this endpoint, but note that when the `frozen` state is

set, but `thawed` not yet, then this implies access restrictions for non-privileged users to other endpoints.

Example

Request:

```
GET https://example.com/api/contests/wf14/state
```

Returned data:

```
{
   "started": "2014-06-25T10:00:00+01",
   "ended": null,
   "frozen": "2014-06-25T14:00:00+01",
   "thawed": null,
   "finalized": null,
   "end_of_updates": null
}
```

Submissions

Submissions, a.k.a. attempts to solve problems in the contest.

The following endpoints are associated with submissions:

Endpoint	Type	Required?	Source @WF	Description
/contests/<id>/submissions	application/json	yes	CCS	JSON array of all submissions with elements as defined in the table below
/contests/<id>/submissions/<id>	application/json	yes	CCS	JSON object of a single submission with elements as defined in the table below

JSON elements of submission objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the submission. Usable as a label, typically a low incrementing number
language_id	ID	yes	no	CCS	identifier of the language submitted for
problem_id	ID	yes	no	CCS	identifier of the problem submitted for
team_id	ID	yes	no	CCS	identifier of the team that made the submission
time	TIME	yes	no	CCS	timestamp of when the submission was made
contest_time	RELTIME	yes	no	CCS	contest relative time when the submission was made
entry_point	string	yes	yes	CCS	code entry point for specific languages
files	array of ARCHIVE	yes	no	CCS	submission files, contained at the root of the archive. Default and only allowed mime type is application/zip.
reaction	array of VIDEO	no	yes	CDS	reaction video from team's webcam.

The **files** attribute provides the file(s) of a given submission as a zip archive. These must be

stored directly from the root of the zip file, i.e. there must not be extra directories (or files) added unless these are explicitly part of the submission content. For **POST**, **PUT** and **PATCH** methods, the **files** attribute must contain the base64-encoded string of the zip archive.

Access restrictions at WF

The **entry_point** and **files** attribute are accessible only for clients with **admin** or **analyst** role. The **reaction** attribute is available to clients with **public** role only when the contest is not frozen.

Example

Request:

```
GET https://example.com/api/contests/wf14/submissions
```

Returned data:

```
[{"id":"187","team_id":"123","problem_id":"10-asteroids",
   "language_id":"1-
java","time":"2014-06-25T11:22:05.034+01","contest_time":"1:22:05.034","entry_point":"Main",
   "files":[{"href":"contests/wf14/submissions/187/files","mime":"application/zip"}]}
]
```

Note that the relative link for `files` points to the location `https://example.com/api/contests/wf14/submissions/187/files` since the base URL for the API is `https://example.com/api`.

Judgements

Judgements for submissions in the contest.

The following endpoints are associated with judgements:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/judgements	application/json	yes	CCS	JSON array of all judgements with elements as defined in the table below
/contests/<id>/judgements/<id>	application/json	yes	CCS	JSON object of a single judgement with elements as defined in the table below

JSON elements of judgement objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the judgement
submission_id	ID	yes	no	CCS	identifier of the submission judged
judgement_type_id	ID	yes	yes	CCS	the verdict of this judgement
start_time	TIME	yes	no	CCS	absolute time when judgement started
start_contest_time	RELTIME	yes	no	CCS	contest relative time when judgement started
end_time	TIME	yes	yes	CCS	absolute time when judgement completed
end_contest_time	RELTIME	yes	yes	CCS	contest relative time when judgement completed
max_run_time	decimal	no	yes	CCS	maximum run time in seconds for any test case

When a judgement is started, each of `judgement_type_id`, `end_time` and `end_contest_time` will be `null` (or missing). These are set when the judgement is completed.

Access restrictions at WF

For clients with the **public** role, judgements will not be included for submissions received while the scoreboard is frozen. This means that all judgements for submissions received before the scoreboard has been frozen will be sent immediately, and all judgements for submissions received after the scoreboard has been frozen will be sent immediately after the scoreboard has been thawed.

Example

Request:

```
GET https://example.com/api/contests/wf14/judgements
```

Returned data:

```
[{"id":"189549","submission_id":"wf2017-32163123xz3132yy","judgement_type_id":"CE","start_time":"201
4-06-25T11:22:48.427+01",
"start_contest_time":"1:22:48.427","end_time":"2014-06-25T11:23:32.481+01","end_contest_time":"1:23:
32.481"},
{"id":"189550","submission_id":"wf2017-32163123xz3133ub","judgement_type_id":null,"start_time":"2014
-06-25T11:24:03.921+01",
   "start_contest_time":"1:24:03.921","end_time":null,"end_contest_time":null}
 ]
```

Runs

Runs are judgements of individual test cases of a submission.

The following endpoints are associated with runs:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/runs	application/json	yes	CCS	JSON array of all runs with elements as defined in the table below
/contests/<id>/runs/<id>	application/json	yes	CCS	JSON object of a single run with elements as defined in the table below

JSON elements of run objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the run
judgement_id	ID	yes	no	CCS	identifier of the judgement this is part of
ordinal	ORDINAL	yes	no	CCS	ordering of runs in the judgement. Must be different for every run in a judgement. Runs for the same test case must have the same ordinal. Must be between 1 and `problem:test_data_count`.
judgement_type_id	ID	yes	no	CCS	the verdict of this judgement (i.e. a judgement type)
time	TIME	yes	no	CCS	absolute time when run completed
contest_time	RELTIME	yes	no	CCS	contest relative time when run completed
run_time	decimal	no	no	CCS	run time in seconds

Access restrictions at WF

For clients with the **public** role, runs will not be included for submissions received while the scoreboard is frozen. This means that all runs for submissions received before the scoreboard has been frozen will be sent immediately, and all runs for submissions received after the scoreboard has been frozen will be sent immediately after the scoreboard has been thawed.

Example

Request:

```
GET https://example.com/api/contests/wf14/runs
```

Returned data:

```
[{"id":"1312","judgement_id":"189549","ordinal":28,"judgement_type_id":"TLE",
  "time":"2014-06-25T11:22:42.420+01","contest_time":"1:22:42.420"}
]
```

Clarifications

Clarification message sent between teams and judges, a.k.a. clarification requests (questions from teams) and clarifications (answers from judges).

The following endpoints are associated with clarification messages:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/clarifications	application/json	yes	CCS	JSON array of all clarification messages with elements as defined in the table below
/contests/<id>/clarifications/<id>	application/json	yes	CCS	JSON object of a single clarification message with elements as defined in the table below

JSON elements of clarification message objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the clarification
from_team_id	ID	yes	yes	CCS	identifier of team sending this clarification request, `null` if a clarification sent by jury
to_team_id	ID	yes	yes	CCS	identifier of the team receiving this reply, `null` if a reply to all teams or a request sent by a team
reply_to_id	ID	yes	yes	CCS	identifier of clarification this is in response to, otherwise `null`
problem_id	ID	yes	yes	CCS	identifier of associated problem, `null` if not associated to a problem
text	string	yes	no	CCS	question or reply text
time	TIME	yes	no	CCS	time of the question/reply
contest_time	RELTIME	yes	no	CCS	contest time of the question/reply

Note that at least one of `from_team_id` and `to_team_id` has to be `null`. That is, teams cannot send messages to other teams.

Access restrictions at WF

Clients with the **public** role can only view clarifications replies from the jury to all teams, that is, messages where both `from_team_id` and `to_team_id` are `null`.

Examples

Request:

```
GET https://example.com/api/contests/wf14/clarifications
```

Returned data:

```
[{"id":"wf2017-1","from_team_id":null,"to_team_id":null,"reply_to_id":null,"problem_id":null,
  "text":"Do not touch anything before the contest
starts!","time":"2014-06-25T11:59:27.543+01","contest_time":"-0:15:32.457"}
]
```

Request:

```
GET https://example.com/api/contests/wf14/clarifications
```

Returned data:

```
[{"id":"1","from_team_id":"34","to_team_id":null,"reply_to_id":null,"problem_id":null,
  "text":"May I ask a question?","time":"2017-06-25T11:59:27.543+01","contest_time":"1:59:27.543"},
 {"id":"2","from_team_id":null,"to_team_id":"34","reply_to_id":"1","problem_id":null,
  "text":"Yes you may!","time":"2017-06-25T11:59:47.543+01","contest_time":"1:59:47.543"}
]
```

Request:

```
GET https://example.com/api/contests/wf14/clarifications
```

Returned data:

```
[{"id":"1","from_team_id":"34","text":"May I ask a
question?","time":"2017-06-25T11:59:27.543+01","contest_time":"1:59:27.543"},
 {"id":"2","to_team_id":"34","reply_to_id":"1","text":"Yes you
may!","time":"2017-06-25T11:59:47.543+01","contest_time":"1:59:47.543"}
]
```

Awards

Awards such as medals, first to solve, etc.

The following endpoints are associated with awards:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/awards	application/json	no	CCS	JSON array of all awards with elements as defined in the table below
/contests/<id>/awards/<id>	application/json	no	CCS	JSON object of a single award with elements as defined in the table below

JSON elements of award objects:

Name	Type	Required?	Nullable?	Source @WF	Description
id	ID	yes	no	CCS	identifier of the award.
citation	string	yes	no	CCS	award citation, e.g. "Gold medal winner"
team_ids	array of ID	yes	no	CCS	JSON array of team ids receiving this award. No meaning must be implied or inferred from the order of IDs. The array may be empty.

Access restrictions at WF

For clients with the **public** role, awards will not include information from judgements of submissions received after the scoreboard freeze until it has been unfrozen.

Semantics

- Awards are not final until the contest is.
- If an award is not intended to be updated during the contest it should not be present during the contest. E.g. if "winner" will not be updated with the current leader during the contest, it should not be **create**d until the award **is** awarded.
- If an award is present during the contest this means that if the contest would end immediately and then become final, that award would be final. E.g. the "winner" during the contest should be the current leader. This is of course subject to what data the client can see; the public role's winner may not change during the scoreboard freeze but an admin could see the true current winner.
- If it's not possible to yet determine how the award would end up if the contest would end immediately, it must not be set. E.g. there should be no team ID for the "first-to-solve-a" award if there are still submissions without a judgement on problem A that are earlier than all accepted submissions on problem A.

For some common award cases the following IDs should be used.

ID	Meaning during contest	Meaning when contest is final	Comment
winner	Current leader(s). Empty if no team has scored.	Winner(s) of the contest	
gold-medal	Teams currently placed to receive a gold medal. Empty if no team has scored.	Teams being awarded gold medals	
silver-medal	Teams currently placed to receive a silver medal. Empty if no team has scored.	Teams being awarded silver medals	
bronze-medal	Teams currently placed to receive a bronze medal, assuming no extra bronze are awarded. Empty if no team has scored.	Teams being awarded bronze medals	
first-to-solve-<id>	The team(s), if any, that was first to solve problem <id>. This implies that no unjudged submission made earlier remains.	Same.	Will never change once set, except if there are rejudgements.
group-winner-<id>	Current leader(s) in group <id>. Empty if no team has scored.	Winner(s) of group <id>	
organization-winner-<id>	Current leader(s) of organization <id>. Empty if no team has scored.	Winner(s) of organization <id>	Not useful in contest with only one team per organization (e.g. the WF)

Example

Request:

```
GET https://example.com/api/contests/wf14/awards
```

Returned data:

```
[{"id":"gold-medal","citation":"Gold medal winner","team_ids":["54","23","1","45"]},
 {"id":"first-to-solve-a","citation":"First to solve problem A","team_ids":["45"]},
 {"id":"first-to-solve-b","citation":"First to solve problem B","team_ids":[]}
]
```

Scoreboard

Scoreboard of the contest.

Since this is generated data, only the **GET** method is allowed here, irrespective of role.

The following endpoint is associated with the scoreboard:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/scoreboard	application/json	yes	CCS	JSON object with scoreboard data as defined in the table below

Scoreboard request options

The following options can be passed to the scoreboard endpoint.

Scoreboard at the time of a given event

By passing an event ID with the "after_event_id" URL argument, the scoreboard can be requested as it was directly after the specified event:

```
/scoreboard?after_event_id=xy1234
```

This makes it possible for a client to get the scoreboard information that is guaranteed to match a certain contest event. In case no "after_event_id" argument is provided, the current scoreboard will be returned.

A suggested efficient server-side implementation to provide this, is to store with each event that changes the scoreboard, the new team scoreboard row.

Scoreboard format

JSON elements of the scoreboard object.

Name	Type	Required?	Nullable?	Source @WF	Description
event_id	ID	yes	no	CCS	Identifier of the event after which this scoreboard was generated. This must be identical to the argument `after_event_id`, if specified.
time	TIME	yes	no	CCS	Time contained in the associated event. Implementation defined if the event has no associated time.
contest_time	RELTIME	yes	no	CCS	Contest time contained in the associated event. Implementation defined if the event has no associated contest time.
state	object	yes	no	CCS	Identical data as returned by the contest state endpoint. This is provided here for ease of use and to guarantee the data is synchronized.
rows	JSON array of JSON objects	yes	no	CCS	A list of rows of team with their associated scores.

The scoreboard `rows` array is sorted according to rank and alphabetical on team name within identically ranked teams. Each JSON object in the rows array consists of:

Name	Type	Required?	Nullable?	Source @WF	Description
rank	integer	yes	no	CCS	rank of this team, 1-based and duplicate in case of ties
team_id	ID	yes	no	CCS	identifier of the team
score	object	yes	no	CCS	JSON object as specified in the rows below (for possible extension to other scoring methods)
score.num_solved	integer	yes	no	CCS	number of problems solved by the team
score.total_time	integer	yes	no	CCS	total penalty time accrued by the team
problems	array of objects	yes	no	CCS	array of problems with scoring data, see below for the specification of each element

Each problem object within the scoreboard consists of:

Name	Type	Required?	Nullable?	Source @WF	Description
problem_id	ID	yes	no	CCS	identifier of the problem
num_judged	integer	yes	no	CCS	number of judged submissions (up to and including the first correct one)
num_pending	integer	yes	no	CCS	number of pending submissions (either queued or due to freeze)
solved	boolean	yes	no	CCS	whether the team solved this problem
time	integer	depends	no	CCS	minutes into the contest when this problem was solved by the team. Required iff `solved=true`

Access restrictions at WF

For clients with the **public** role, the scoreboard will not include information from judgements of submissions received after the scoreboard has been frozen until it has been thawed.

Example

Request:

```
GET https://example.com/api/contests/wf14/scoreboard
```

Returned data:

```
{
  "event_id": "xy1234",
  "time": "2014-06-25T14:13:07.832+01",
  "contest_time": "4:13:07.832",
  "state": {
    "started": "2014-06-25T10:00:00+01",
    "ended": null,
    "frozen": "2014-06-25T14:00:00+01",
    "thawed": null,
    "finalized": null,
    "end_of_updates": null
```

```
  },
  "rows": [
    {"rank":1,"team_id":"123","score":{"num_solved":3,"total_time":340},"problems":[
      {"problem_id":"1","num_judged":3,"num_pending":1,"solved":false},
      {"problem_id":"2","num_judged":1,"num_pending":0,"solved":true,"time":20},
      {"problem_id":"3","num_judged":2,"num_pending":0,"solved":true,"time":55},
      {"problem_id":"4","num_judged":0,"num_pending":0,"solved":false},
      {"problem_id":"5","num_judged":3,"num_pending":0,"solved":true,"time":205}
    ]}
  ]
}
```

Event feed

Provides the event feed for the current contest. This is effectively a changelog of create, update and delete events taking place in the REST endpoints. Some endpoints (specifically the Scoreboard and the Event feed itself) are aggregated data, and so these will only ever update due to some other REST endpoint updating. For this reason there is no explicit event for these, since there will always be another event sent. This can also be seen by the fact that there is no scoreboard event in the table of events below.

Since this is generated data, only the **GET** method is allowed here, irrespective of role.

The following endpoint is associated with the event feed:

Endpoint	Mime-type	Required?	Source @WF	Description
/contests/<id>/event-feed	application/x-ndjson	yes	CCS	NDJSON feed of events as defined below

Multiple requests of the event feed must return the exact same events in the exact same order, except that events filtered out by the feed options must be left out and new elements, if any, are added in later requests.

The event feed is a streaming endpoint that does not terminate under normal circumstances. To ensure keep alive, if no event is sent in 120 seconds, a newline must be sent.

Feed options

There are options for filtering based on events and starting the feed at a specified event. Any combination of these may be specified.

Filtering events

If a client only wants some types of events the feed can be filtered with the "types" URL argument:

```
/event-feed?types=submissions,teams
```

If not specified all events will be sent. If specified only events of the (comma separated) listed types will be sent.

Feed starting point

If a client wants data from some point in time this can be done with the "since_id" URL argument:

```
/event-feed?since_id=dj593
```

If specified the event feed will include all events strictly *after* the specified id. If a client copies the id of an event and uses that for the id URL argument it will get all events after that event. This is useful e.g. if a client is disconnected and wants to continue where it left off.

If the id is not specified the event feed will include all events from the beginning of the feed. The request will fail with a 400 error if the id is invalid.

Feed format

The feed is served as Newline delimited JSON, with every event as its own JSON object.

This document uses the term "element". An element corresponds to a single object (run, judgement, language, team, etc.).

The general format for events is:

```
{"type": "<event type>", "id": "<id>", "op": "<type of operation>", "data": <JSON data for element>
}
```

Name	Type	Required?	Nullable?	Description
type	string	yes	no	Type of event, one of the events in the table below. Can be used for filtering.
id	ID	yes	no	Unique identifier for the event.
op	string	yes	no	Type of operation, one of **create**, **update**, **delete**.
data	object	yes	no	For **create** and **update**, the object that would be returned if calling the corresponding API endpoint at this time. For **delete** an object with only the **id** attribute with value the identifier of the deleted element.

All event types have a corresponding API endpoint, as specified in the table below.

Event	API Endpoint
contests	/contests/<id>
judgement-types	/contests/<id>/judgement-types/<id>
languages	/contests/<id>/languages/<id>
problems	/contests/<id>/problems/<id>
groups	/contests/<id>/groups/<id>
organizations	/contests/<id>/organizations/<id>
teams	/contests/<id>/teams/<id>
team-members	/contests/<id>/team-members/<id>
state	/contests/<id>/state
submissions	/contests/<id>/submissions/<id>
judgements	/contests/<id>/judgements/<id>
runs	/contests/<id>/runs/<id>
clarifications	/contests/<id>/clarifications/<id>
awards	/contests/<id>/awards/<id>

General access restrictions

The event responses and **data** objects contained in it must observe the same restrictions as those of the respective endpoints they represent. This means that attributes inside the **data** element will be present if and only if the client has access to those at the respective endpoint. The client only receives create, update and delete events of elements it has (partial) access to. When time-based access is granted or revoked, create or delete events are dispatched for each affected entity.

The referential integrity requirement must be strictly adhered to by the event feed. I.e. the API state as defined by the event feed "changelog" must at all points have referential integrity:

- If an object, A, is deleted and another object, B, refers to it, then the event that shows that B has been updated to not refer to A or that B has been deleted *must* come before the event that shows that A has been deleted.
- If some data is only available after a specific state change, then the event showing the state change *must* come before any update events making that data available. E.g. problems are only available after contest start for the public role, so the state event showing that the contest has started *must* come before the problems events creating the problems.
- Since nothing must change after the contest has ended, thawed (or never been frozen) and been finalized, no event may come after the state event showing that.

Example

Request:

```
GET https://example.com/api/contests/wf14/event-feed
```

Returned data:

```
{"type":"teams","id":"k-2435","op":"create","data":{"id":"11","icpc_id":"201433","name":"Shanghai
Tigers","organization_id":"inst123","group_id":"asia"}}
{"type":"teams","id":"k-2436","op":"update","data":{"id":"11","icpc_id":"201433","name":"The
Shanghai Tigers","organization_id":"inst123","group_id":"asia"}}
{"type":"teams","id":"k-2437","op":"delete","data":{"id":"11"}}
```
