Contest Control System Requirements

From ICPC-Contest Control Standard
Jump to:navigation, search

Contents

Introduction

This document specifies requirements for operation, verification and validation of Programming Contest Control Systems that wish to be considered for managing the operation of the ACM International Collegiate Programming Contest World Finals. The document defines a set of operations, capabilities, and features which any candidate system must provide, including a set of testability criteria which must be satisfied and a set of documentation which must be provided.

The current draft of the requirements document is *Version 1.0*, published *<insert publication date>* by the *Competitive Learning Institute* (*<insert CLI website URL>*). [TODO: insert into the previous sentence the final publication date once the requirements has been approved; insert official CLI website URL once that has been created.]

The primary authors of the document are John Clevenger and Fredrik Niemelä, acting under the auspices of the Competitive Learning Institute (CLI). Contributors to the document include Samir Ashoo, Per Austrin, Troy Boudreau, Tim deBoer, Emma Enström, Mikael Goldman, Gunnar Kreitz, Doug Lane, Pehr Söderman, and Mattias de Zalenski.

Overview

For purposes of this requirements document, the term *contest control system* (CCS) means software

which automatically manages the operation of a programming contest. Operations to be managed by the CCS include: running submissions by teams, judging of submissions, handling clarification requests from teams and clarification responses from judges, calculation of standings, generating external representations of contest results, and overall CCS configuration.

A contest control system may be submitted to the Director of Operations for the ICPC World Finals for consideration as a candidate to run the World Finals. The items which must be submitted in order to be considered are described under Certification Process in this document. Any submitted CCS which meets all of the requirements defined in this document will be certified as being accepted as a candidate to run the World Finals. Meeting or failing to meet each requirement will be determined by a test specified by the Director of Operations and/or his/her designee.

The choice of the CCS actually used to run the World Finals each year will be made by the Director of Operations from among those CCSs which have been accepted as candidates prior to the acceptance deadline.

Rationale

This document is intended to specify functional requirements, not implementation requirements. Nothing in this document should be construed as requiring any particular implementation architecture or use of any particular implementation language or languages. For example, a CCS might be implemented as a set of stand-alone applications using a client-server architecture, or might equally well be implemented as a set of scripts, web pages, and/or other facilities.

These requirements are not intended to act as a specification for what constitutes "best practices" in a Contest Control System; rather, it acts solely as a specification of what functionality is required to be a candidate for running the ICPC World Finals. As such, there may be functions listed in this document which are specific solely to the ICPC World Finals but might not be required for running other contests; a candidate CCS must meet the requirements for these functions.

Likewise, there may be functionality which a CCS provides to meet the requirements of some other contest; such additional functionality will not disqualify the CCS from qualifying as a candidate to run the ICPC World Finals provided the CCS meets all the requirements listed in this document.

General Requirements

Licensing

The CCS must either be freely useable, or must be accompanied by a license granting to the ACM-ICPC the non-exclusive, non-revocable, non-transferable rights of use of the CCS both for purposes of evaluation and testing and for use in the ICPC World Finals. The CCS may not require obtaining any third-party license in order to be used by the ACM-ICPC.

Data Persistency

Contest Configuration Persistency

The CCS must support persistence of the contest configuration. This means that following a shutdown of the CCS, or a failure on the machine(s) on which the CCS runs, it must be possible to quickly restore the configuration information already entered into the CCS without the necessity of reentering that configuration data.

Contest State Persistency

The CCS must support persistence of the contest state once the contest has been started. This means that following a shutdown of the CCS, a power or hardware failure on the machine(s) on which the CCS runs, it must be possible to quickly restore the contest state to what it was prior to the disruption. The contest state for purposes of this requirement includes all data regarding team submissions, clarification requests, clarifications, judgements, and similar data defining and/or affecting the standings of the contest.

The CCS must not lose more than 1 minute of contest data on a failure of any part of the contest system.

Account Types

The CCS must support the ability of clients to access the system through different *account types*.

The type of account which a client uses to access the system constrains the functions which the client may perform, as described below.

At least the following different types of accounts must be supported:

- Team: used by teams competing in the contest. The functions available to team clients are as defined in the section on the team interface.
- Judge: used by judges in the contest. The functions available to judge clients are as defined in the section on the judge interface.
- Admin: used by contest administrators. The functions available to admin clients are as defined in the section on the admin interface.

Secure Authentication

The CCS must support a secure authentication mechanism, allowing each registered user (being either a team, an admin, or a judge) to gain access to the contest, and must ensure that:

1. Only users who supply correct authentication credentials may invoke contest-related functions.
2. Users may only invoke functions corresponding to their authentication credentials. (In other words, it must not be possible for, e.g., a team to invoke functions as if they were some other team, or as if they were a judge.)

The CCS may rely on the underlying operating system account login/password mechanism for purposes of meeting the requirements of this section, provided that it is the case that the CCS enforces all of the requirements of this section, including but not limited to the requirement that users not be allowed to invoke functions as if they were some user other than that specified by their login credentials.

Authentication Data

If the CCS uses a login/password mechanism to enforce secure authentication, it must support account creation and password assignment according to the following algorithm:

1. read teams.tsv defining team accounts to be created
2. read accounts.tsv defining additional accounts to be created or, for teams, specifying the password of the accounts.

Logging Out

The CCS must support a mechanism for each user to disconnect from the contest, with the effect that no other user of the computer system not possessing that team's authentication credentials will be able to invoke CCS functions as if they did possess those credentials.

Network Security

All communication between CCS modules which takes place over the contest network must be encrypted.

No outside contact

The CCS must be able to run at the World Finals site without contact to anything outside the contest network.

Timestamps and IDs

A timestamp and an integer sequence, called an ID, are assigned to submissions and clarification requests at the time they enter the system. The following constraints must be enforced by the CCS:

- Submissions received earlier must have a lower ID than submissions received later. The same holds for the IDs of clarification requests.
- Timestamps must have at least second granularity.

Limitations

Number of Problems

The CCS must support at least 15 problems. Specifically any UI, scoreboards or other reports should be fully useable with that number of problems.

Number of Teams

The CCS must support at least 200 teams.

Test Data Files

The CCS must support at least 200 test data files per problem, a maximum size of 8GB per test data file and a combined size for all test data files in a problem of 20GB.

Use At The World Finals

In order for a CCS to be selected for use at the World Finals the following constraints must be satisfied:

Support Personnel

At least one person from the submitting entity with intimate knowledge about the inner workings of the CCS must be willing and able to attend at the World Finals where it is used.

Computer Platform

The CCS must run on the computer platform set aside for the CCS at the World Finals and posted at the ICPC web site. Normally this will consist of one machine per team, several auto-judging machines that are identical to the team machines, and one or more servers.

Contest Configuration

Advance Configuration

All variable aspects of the CCS must be configurable prior to the start of a World Finals contest. That is, the CCS may not require that any changes be made to the CCS configuration once the contest has started. Note that this does not alter any requirements which state that the CCS must allow certain configuration changes to be able to be made after the contest starts; it means that the contest administration staff must not be required to make any changes once the contest starts.

Importing Contest Configuration

The CCS must be able to import contest configuration from a single URL, and use the data at that location to configure the CCS. The configuration URL contains all contest configuration data, including the contest configuration file (contest.yaml), problemset configuration file (problemset.yaml), problem definitions, teams.tsv, groups.tsv and logos.tar.gz. For example, if the URL is http://10.1.1.0/data, teams.tsv can be imported from http://10.1.1.0/data/teams.tsv.

The CCS must be able to read teams.tsv, which contains a list of teams registered for the World Finals. It must use the data in this file to automatically create whatever internal representation is necessary to allow each team (and only the teams) specified in the registration list to participate in the World Finals contest.

The CCS must be able to read groups.tsv, an initialization file containing the list of groups to which the teams belong.

Clock Synchronization

The CCS must be able to synchronize its clock to an NTP server provided on the contest network.

Predefined Clarification Answers

The CCS must be able to configure predefined answers to clarification requests (e.g., "No response, read problem statement." and "This will be answered after the practice session.") from the contest configuration file, so that judges can choose to reply to a clarification request by selecting a predefined answer rather than being required to enter a specific answer. One of the predefined answers must be the "default" answer.

Clarification Categories

The CCS must be able to configure "categories" to which clarification requests can be assigned from the contest configuration file. A request belongs to exactly one category. Examples of categories are "General", "SysOps", "Operations".

In addition, the CCS must construct one category per problem, i.e., categories named e.g. "Problem A", "Problem B", etc. for each problem.

Programming Languages

Supported Languages

The CCS must provide the ability to compile and execute (or interpret, as appropriate for the language) submitted source code files for each of the languages specified by the Environment of the World Finals.

Language Options

For each supported language compiler or interpreter it must be possible to configure the CCS to invoke it with any of the options specified in the compiler or interpreter's documentation.

Contest Problems

Problems (including judge data, validators, execution time limit, etc) are specified and configured using the Problem format. The CCS must support this format with the following exceptions:

- The CCS does not have to support the use of the following keys in problem.yaml: type, libraries or languages. This means that the default value will be used.
- The CCS does only have to support the values "default" or "custom" for the key validation in problem.yaml.
- The CCS does not have to support sections 5 "Included Code", 9 "Graders" or 10 "Generators".

The CCS must report an error when importing problems that use any unsupported features. The CCS may report an error when unsupported keys are used, even if they are given the default value.

The problem name is as defined in the *problemname* macro in the problem specification. Optionally, the *plainproblemname*, if present, may be used instead. If *problemname* is used then escaped parts of it must be handled properly. If *plainproblemname* is used it must be used verbatim.

Configuration Change

The CCS must allow updating of configuration data without restarting or stopping the CCS.

The CCS must support the ability to create and save an updated contest.yaml file upon request.

Admin Interface

This section describes all the required capabilities for users authenticated as admin.

Account Disabling

The CCS must have a mechanism to disable any account (either an account for a human user or for another system), without the need for starting or stopping the contest. For example, this includes team accounts and judge accounts.

Changes To Authentication Data

The CCS must allow user authentication credential information to be changed dynamically by contest administration staff while the contest is running.

Starting the Contest

The contest must automatically start when the configured start time is reached. It must also be possible to start the contest at the current time.

Adjusting for Exceptional Circumstances

Removing time intervals

It must be possible to, potentially retroactively, specify time intervals that will be disregarded for the purpose of scoring. The time during all such intervals will not be counted towards a team's penalty time for solved problems. Beginning and end of time intervals are given in wall-clock time.

Note that removing a time interval changes the wall-clock time when the contest ends, as the duration of the contest in contest.yaml is specified in contest time.

Removing the interval between time T_0 and T_1, where $T_0 \leq T_1$, means that all submissions received between T_0 and T_1 will have the same contest time and that the time for all submissions received after T_1 will have a contest time that is T_1-T_0 lower than if the interval was not removed.

The removal of a time interval must be reversible. This means that if a time interval is removed, the CCS must support the capability of subsequently restoring the contest to the state it had prior to the removal of the time interval.

Note that time interval removal does not affect the order in which submissions arrived to the CCS. If submission S_i arrived before submission S_j during a removed interval, S_i must still be considered by the CCS to have arrived strictly before S_j.

Changing contest length

It must be possible to change the length of the contest at any time during the contest.

Adding penalty time

It must be possible to specify, for each team, an integer, potentially negative, amount of penalty time to be added into that team's total penalty time.

Ignoring submissions

It must be possible to remove, ignore or somehow mark a submission so that it in no way affects the scoring of the contest.

Pausing Judging

The CCS must provide some way to temporarily pause judging on a per problem basis. While judging is paused teams should still be able to make submissions and these submission should be shown on the scoreboard as usual.

Rejudging

Automatic Rejudging

The CCS must allow the ability to automatically rejudge a selected set of submissions.

Each submission in the set of selected submissions is executed and judged in the same manner as for a newly arrived submission.

Previewing Rejudgement Results

There must be a way to preview the judgements which result from rejudging the set of selected submissions without committing the resulting judgements.

Submission Selection

The CCS must provide the ability to specify a filter defining the set of submissions to be rejudged. The CCS must support any combination of filters of the following types:

1. A specific (single) submission.
2. All submissions for a specific problem.
3. All submissions using a specific language.
4. All submissions by a specific team or set of teams.
5. All submissions between some time T_0 and some subsequent time T_1.
6. All submissions which have been assigned any specific one of the allowable submission judgments as defined in Judge Responses, or all submissions that received any judgment other than "Accepted" (that is, all rejected submissions).
7. All submissions which have been run on a specific computer (identified in some reasonable way, e.g., IP address or hostname). This requirement is only applicable if the CCS uses multiple machines to run submissions.

Thus, for example, it must be possible to select "all rejected submissions for problem B", "all Time Limit Exceeded submissions using Java for problem C", "all submissions between time 2013-07-01 08:00:00+00 and time 2013-07-01 09:00:00+00 of the contest using Java", or "all submissions using C++".

Manual Override of Judgments

The CCS must support the ability to assign, to a single submission, an updated judgment chosen from among any of the allowed submission judgments as defined in Judge Responses.

The CCS must require a separate authentication every time a judgment is changed manually and all such changes must be logged.

Scoreboard Display

The CCS must provide a mechanism for judges to view the current scoreboard. The scoreboard must be updated in such a way that it's never more than 30 seconds out of date.

During times when the scoreboard is frozen, administrators must be able to view the current (updated) scoreboard as well as the frozen scoreboard.

Freezing the Scoreboard

The scoreboard must automatically freeze when the configured scoreboard freeze time is reached. It must also be possible to manually freeze the scoreboard at the current time. All submissions received after the freeze time must be treated as pending on a frozen scoreboard.

The exact phrase displayed on the frozen scoreboard must be:

```
 The scoreboard was frozen with XX minutes remaining - submissions in the
last XX minutes of the contest are still shown as pending.
```

where XX is the number of minutes remaining in the contest at the time the scoreboard was frozen.

It must be possible to re-enable scoreboard display updating at any time after it has been disabled, again without stopping the contest or affecting contest operations in any way.

Finalizing the Contest

Finalizing is the procedure to authorize the final results at the end of a contest. The results.tsv and scoreboard.tsv files will be generated and the finalized element will be sent on the Event Feeds.

When the contest is over, but not finalized, all scoreboards must show a warning that the results are not final.

The CCS must provide a way for admins to finalize the contest. It must *not* be possible to finalize a contest if one or more of the following applies:

1. The contest is still running (i.e., the contest is not over).
2. There are un-judged submissions.
3. There are submissions judged as Judging Error.
4. There are unanswered clarification requests.

The following fields need to be entered before Finalizing the contest:

1. B, a non-negative integer, as used in Scoring Data Generation.
2. a comment, string, that indicates who approves the Final Results, for Example "Finalized by John Doe and Jane Doe".

Team Interface

This section describes all the required capabilities for users authenticated as team.

Submissions

For purposes of this document, solutions to problems submitted for judging are called **submissions**. A submission consists of a set of source code files sent as a single unit at one time to the judging system by a team.

Submission Mechanism

The CCS must provide each team with the ability to make a submission to the judging system.

Submission Contents

A team must be able to specify, for each submission:

1. the contest problem to which the submission applies;
2. the programming language used in the submission;
3. the source code file or files comprising the submission.

The CCS must allow teams to specify at least 10 files in a given submission and must allow teams to

make submissions for any defined contest problem, written in any defined contest programming language.

Team Viewing of Submission Status

The CCS must provide each team with a capability for reviewing the status of each submission the team has made, including: the contest time of the submission, the language and problem specified in the submission; and the most recent judgment (if any) for the submission.

Submission Security

The CCS must ensure that no team can learn anything about the submissions of any other team (other than what can be deduced from the scoreboard).

Clarification Requests

A clarification request is a message sent from a team to the judges asking for clarification regarding a contest problem or possibly the contest in general.

Sending Clarification Requests

The CCS must provide each team with the ability to submit a clarification request to the judges over the network.

Content of Clarification Requests

The team must be able to specify the text content and category of a clarification request.

Viewing of Clarification Request Status

The CCS must provide each team with a capability for reviewing the status of each clarification request the team has submitted, including: the contest time of the clarification request; the problem identified in the clarification request if identification of a specific problem was required by the CCS; and the response from the judges to the clarification request if any response has occurred yet.

Clarification Request Security

The CCS must ensure that no team can see the clarification requests of any other team, except as provided in the section Judge Interface.

Broadcast Messages

Team Viewing of Broadcast Messages

The CCS must provide each team with a capability for viewing any broadcast messages sent by the judges (see Issuing Broadcast Messages under Judging).

Notifications

The CCS must notify teams when a judgement, clarification or broadcast message has been received. This notification may not steal focus.

Scoreboard Display

The CCS must provide a mechanism for teams to view the current scoreboard. The scoreboard must be updated in such a way that it's never more than 30 seconds out of date.

Judge Interface

This section describes all the required capabilities for users authenticated as judge.

Simultaneous Usage

It must be possible for multiple human judges, working on different computers, to simultaneously perform the operations specified in this subsection (on different submissions).

Viewing Submissions

The CCS must provide a human judge with the ability to perform each of the following operations:

1. See a list of all submissions, where the list includes (for each submission) the contest time at which the submission was sent, the problem and language specified in the submission, and any judgments applied.
2. Sort the list of submissions by submission time (newest submissions first).
3. Filter the list of submissions by:
    1. Problem
    2. Team
    3. Language
    4. Judgement applied.
4. View and download the output produced by the submission when run against the specified input data.
5. View and download the source code contained in any specific submission.
6. View the compiler output resulting from compiling any specific submission using the compiler arguments configured in the CCS.
7. View the validator output resulting from invoking the external validator associated with the contest problem for any specific submission.
8. View and download the judge's input data file associated with the contest problem for any specific submission.
9. View and download the "judge's output" (the "correct answer" file) associated with the contest problem for any specific submission.
10. View the judge data description, if available.
11. View a diff between team output for the submission and judge answer file for the problem.
12. View previous submissions by the same team on the same problem.

In addition, any additional analytical capabilities allowing the judges to track differences between submissions are appreciated.

Handling Clarifications Requests

The CCS must provide a human judge with the ability to perform each of the following operations:

1. See a list of all clarification requests, where the list includes (for each clar) the team which submitted the request, the contest time at which the clar was submitted, and an indication of whether or not an answer to the clar has been sent to the team which submitted it.

2. Sort the list of clarification requests by time.
3. Filter the list of clarification requests by:
   1. A user-specified set of categories
   2. A team
   3. Whether the clarification request has been answered.
4. Determine, for any specific clarification request, what answer was returned to the team if the clar has already been answered.
5. Compose an answer to the clar and send it, along with the text of the original clarification request, to the team.
6. Optionally choose to also send the clarification request text and answer to all teams in the contest.
7. Change the category of a clarification request.

Issuing Broadcast Messages

The CCS must provide the ability for a human judge to compose a message and broadcast that message to all teams in the contest. It must be possible to do this even when the contest is not running.

Scoreboard Display

The CCS must provide a mechanism for judges to view the current scoreboard. The scoreboard must be updated in such a way that it's never more than 30 seconds out of date.

During times when the scoreboard is frozen, judges must be able to view the current (updated) scoreboard as well as the frozen scoreboard.

Judging

Automated Judgements

The CCS must be able to automatically judge incoming submissions. This mechanism must automatically (that is, without human intervention) process each incoming submissions, and for each such submission must automatically:

1. Compile (if appropriate for the language) the program contained in the submission, enforcing the compilation time limit.
2. Execute the program contained in the submission, with the corresponding contest problem data automatically supplied to the program.
3. Prevent the submitted program from performing any prohibited operations.
4. Enforce any configured execution time limit, memory limit, and output size limit specified for the corresponding problem.
   - The execution time limit gives a restriction on the amount of *CPU time* that the submission may consume, per test file.
   - In addition, the CCS must restrict the amount of *wall clock time* that the submission may consume (to safeguard against submissions which spend a long time without using CPU time by e.g., sleeping).
5. Invoke an external program, known for purposes of this document as a validator, passing to it the output generated by the program specified in the submission and getting back from it an indication of what judgment is to be applied to the submission (see the External Validators section).

6. Assign an appropriate judgment to the submission.
7. Send a notification about the judgement to the team.

It must be possible to configure the CCS such that these actions are performed on a machine that is not accessible to any team (possibly just a different virtual machine, e.g. if the machines used in the contest are thin clients connecting to a shared server).

Prevention of Auto-judge Machine Starvation

The CCS must use auto-judge machines efficiently and fairly. At a minimum it needs to ensure:

1. Submissions must not be left in queue if there are unused auto-judge machines.
2. The system must prevent a single team from starving other teams out of auto-judge machines.

Prohibited Operations

The CCS must ensure that prohibited operations in submissions have no non-trivial outside effects.

The prohibited operations are:

1. Using libraries except those explicitly allowed
2. Executing other programs
3. Reading any files
4. Creating files
5. Sending signals to other programs
6. Side-stepping time or memory limits
7. Sending or receiving network traffic, e.g. opening sockets

Judge Responses

The CCS must answer each submission with a responses from the table below. Some judgements results in penalty time being added and only Accepted means that the problem should be counted as solved, as specified in the table.

Response	Acronym	Penalty	Solved
Compile Error	CE	No	No
Run-Time Error	RTE	Yes	No
Time Limit Exceeded	TLE	Yes	No
Wrong Answer	WA	Yes	No
Accepted	AC	No	Yes
Security Violation	SV	Yes	No
Judging Error	JE	No	No

Assigning a Judgment

The next two sections define how to assign a judgment to a submission for problems with a single input file and with multiple input files, respectively. Note however that the **Judging Error** and **Security Violation** judgments constitute exceptions to this, as defined in Exceptional Judgments.

Judging With a Single Input File

To determine which answer to use, the following rules must be applied in order:

1. If the submitted program fails to compile or compilation exceeds the compilation time limit, the response must be **Compile Error**.
2. If the submitted program exceeds the memory limit or crashes before the execution time limit is exceeded, the answer must be **Run-Time Error**.
3. If the submitted program runs longer than the execution time limit, the answer must be **Time Limit Exceeded**.
4. If the output of the submitted program exceeds the output size limit or if the output of the submitted program is not accepted by the output validator, the answer must be **Wrong Answer**.
5. If the output of the submitted program is accepted by the output validator, the answer must be **Accepted**.

Judging With Multiple Input Files

If the problem has multiple judge input files the judgment is assigned as follows:

1. For each input file apply the decision process for a single input file.
2. If any file is not judged as **Accepted**, the response must be that of the first file, in alphabetical order, that was not judged **Accepted**.
3. Otherwise the response must be **Accepted**.

Note that the CCS is only required to judge as many files as needed to determine the first file, if any, that is not judged **Accepted**.

Exceptional Judgments

The preceding sections define how to assign a judgment to a submitted program. However, the following two exceptions apply:

1. If, during any point of the judging process an error occurs that the CCS can not recover from, **Judging Error** must be the judgment.
2. If, during any point of the judging process the submitted program tries to perform a prohibited operation, **Security Violation** may be the judgment.

Validators

A CCS fulfilling these requirements must safeguard against faulty validators. For instance, if a validator were to produce excessively large feedback files, or crash, the CCS must handle this gracefully and report it to contest staff. Reasons for such misbehaviour of the validator program could be for instance a security bug in the validator program, enabling malicious submissions to produce feedback files of their own choosing.

The content of *stdout* and *stderr* of the output validator can be ignored by the contest control system.

Scoring

Scoring Data Generation

The CCS must be capable of automatically generating up-to-date scoring data according to the following:

1. For purposes of scoring, the *contest time of a submission* is the number of minutes elapsed from

the beginning of the contest when the submission was made, skipping removed time intervals if specified (see Removing Time Intervals). This is rounded *down* to the nearest minute, so 59.99 seconds is 0 minutes.
2. The *contest time that a team solved a problem* is the contest time of the team's first accepted submission to that problem.
3. A team's *penalty time on a problem* is the contest time that the team solved the problem, plus *penaltytime* (from contest.yaml) minutes for each previous submission rejected with a judgement that causes penalty time, by that team on that problem, or 0 if the team has not solved the problem.
4. A team's *total penalty time* is the sum of the penalty times for all problems plus any judge added penalty time.
5. A team's *last accepted submission* is the contest time of the problem that the team solved last.
6. The *position* of a team is determined by sorting the teams first by number of problems solved (descending), then within that by total penalty time (ascending), then within that by last accepted submission (ascending).
7. The *rank* of a team is then determined as follows:
    1. For teams in positions up to and including 12+B, the rank equals the position (B is provided when finalizing the contest; the default value is 0).
    2. Teams that solved fewer problems than the median team are not ranked at all.
    3. For the remaining teams, the rank is determined by sorting the teams by number of problems solved (descending).
8. The *award* of a team is:
    1. *gold* if rank is 1 through 4.
    2. *silver* if rank is 5 through 8.
    3. *bronze* if rank is 9 through 12+B.
    4. *ranked* if rank is not 1 through 12+B but the team is ranked.
    5. *honorable* otherwise.

When a number of teams are tied for the same position/rank, they all occupy the same position/rank and a suitable number of subsequent positions/ranks are left empty. For instance, if four teams are tied for 17th position, they are all in 17th position and positions 18, 19 and 20 are unoccupied.

A submission is pending judgement if it has no judgement or if the judgement is **Judging Error**. Pending judgements have no effect on scoring.

Scoreboard

The *current scoreboard* lists the teams sorted by position (with alphabetical order on university name as tie breaker).

The scoreboard must include at least the following information.

For each team:

1. university name
2. univerisity logo (see logos.tar.gz)
3. team position
4. number of problems solved
5. total penalty time

For each team and problem:

1. number of submissions from that team on that problem,
2. whether the problem is solved, unsolved or a judgement is pending,
3. if the problem is solved, the contest time at which it was solved,
4. if the problem is pending judgement, how many submissions are pending judgement on that problem,

A problem is pending judgement if any submission on it is pending judgement and there is not an earlier accepted submission on that problem.

Data Export

Event Feed

It is a requirement that the CCS provide an *external event feed*. This means that the CCS must have a mechanism for external processes to connect to the CCS and obtain dynamic real-time updates regarding the current state of the contest. The CCS event feed mechanism must comply with the Event Feed specification.

Scoreboard Data File

The CCS must be capable of generating an external file containing the current scoreboard. The format of this file must be as defined in scoreboard.tsv.

The CCS must automatically save an updated copy of the external scoring data file whenever the scoring data itself is updated as described above.

Final Results Data File

The *final results* lists the ranked teams sorted by rank (with alphabetical order on team name as tie breaker), followed by the unranked teams (sorted alphabetically by team name). It includes for each ranked team their rank and:

1. The number of problems solved, if the team is ranked (i.e., if they solved more than the median number of problems).
2. The total penalty time and time of last accepted submission, if the team is in rank 1 through 12+B (where B is as defined in Scoring Data Generation).

The CCS must be capable of generating an external file containing the final results of the World Finals contest. The format of this file must be as defined in the results.tsv.

Shadow Mode

Overview

Each World Finals contest is managed by a single CCS, called the *Primary CCS*. Participants in a WF contest (teams, judges, contest administrators, etc.) connect only to the Primary CCS; all participant-visible contest operations are handled by the Primary CCS and the final results of the contest are computed by the Primary CCS.

In order to insure accuracy of the final contest results, each World Finals also runs an independent "parallel" CCS called the *Shadow CCS*. The main task of the Shadow CCS is to verify that the final results computed by the Primary CCS are correct according to the rules of the World Finals. A

contest control system acting as a Shadow CCS is said to be running in *shadow mode*. This section defines requirements which must be implemented by a CCS related to shadow mode.

Support for both Primary and Shadow Modes

A CCS must be able to operate in either Primary or Shadow mode. This means that it must in principle be possible to run two identical instances of the CCS, one set to run in Primary mode and the other set to run in Shadow mode, and where the two CCS instances communicate with each other as defined by the requirements of this section and where each CCS instance performs all of the required operations of a Primary or Shadow CCS respectively. A CCS will not be considered as a candidate for use at the World Finals unless it meets all of the requirements specified in this document, including all of the requirements listed in this section. That is, a CCS will not be considered as a candidate for use at the World Finals unless it is capable of operating both as a Primary CCS and as a Shadow CCS.

The requirement for the ability to run in either Primary mode or Shadow mode does not imply that a CCS must be able to operate in these modes simultaneously, nor that a CCS must be able to switch between these modes dynamically during a contest. Rather, it means that in order to be eligible to be a candidate for use at the World Finals, a CCS must support BOTH Primary and Shadow-mode operations and must meet all of the requirements for both modes, regardless of which mode it would actually be used in for the particular World Finals.

External Communication

Beyond communicating with its own components (e.g., communicating with its external "auto-judges" or with the humans managing it), the only external communication which a CCS running in shadow mode is allowed to require is that it has the ability to communicate with at least one "server" component of the Primary CCS. In particular, a CCS running in shadow mode may not require that it be able to communicate directly with any of the actual contest participants (meaning the teams, judges, or contest administrators who communicate with the Primary CCS).

Said another way, the Shadow CCS must be able to perform all of its functions while communicating only with the Primary CCS.

CCS Configuration

A CCS running in shadow mode must be able to import a complete contest configuration as described in the Importing Contest Configuration section of this Requirements Specification. In particular a shadow-mode CCS must be able to import a contest configuration from a single URL and must also be able to import the teams.tsv and groups.tsv files as specified in that section.

Submissions

A CCS running in shadow mode must be able to obtain team submissions via communication with the Primary CCS. As required elsewhere in this specification, the Primary CCS must implement the REST endpoints defined in the CLICS CCS REST interface specification. The following requirements then apply to the mechanisms used by the Shadow CCS for obtaining submissions:

1. The Shadow CCS should use the **/event-feed** REST endpoint defined in the CLICS CCS REST interface specification to obtain notification of team submissions from the Primary CCS (in other words, the Shadow CCS should preferably use event-driven operation, rather than polling, for team submission notification).

2. The Shadow CCS must use the **/submissions/<id>** and **/submissions/<id>/files** REST endpoints defined in the CLICS CCS REST interface specification for obtaining submission data from the Primary CCS.
3. The Shadow CCS must provide an interface that can display information regarding any submission by looking up the submission using the submission id assigned by the Primary CCS.
4. The Shadow CCS must use the timestamp assigned to a submission by the Primary CCS as the time of the submission for purposes of computing standings, regardless of the time at which the submission is actually obtained by the Shadow CCS.
5. A submission obtained from the Primary CCS must be executed and judged by the Shadow CCS in the same manner as if it had been submitted to the CCS running in non-shadow mode.

Results

A CCS running in shadow mode must produce the same set of output files as those required of a Primary CCS -- that is, it must produce at least the scoreboard.tsv and results.tsv files as defined elsewhere in this Requirements Specification.

A CCS running in shadow mode must in addition provide the capability to present a “diff” of submissions -- that is, a list of all submissions which at the current time have a different judgement than that which was assigned by the Primary CCS.

Requirements Which May Be Imposed By The CCS

A CCS may require the following conditions to exist in order to guarantee support for shadow-mode operations:

- The Shadow CCS is provided with “auto-judging” hardware which is the same as that being used by the Primary CCS.
- The Shadow CCS has a network connection to both the Event Feed and the REST services provided by the Primary CCS.
- The Shadow CCS is provided with credentials which allow appropriate access to the required Primary CCS REST endpoints.
- SubmissionIDs for submissions are unique contest-wide, even in the event of a contest with multiple independent sites being managed by a single distributed Primary CCS.
- The Shadow CCS has access to the same contest configuration files as those used to configure the Primary CCS.

A CCS may not require any conditions not specified above to exist in order to guarantee support for shadow-mode operations.

Documentation Requirements

In order for a given CCS to be considered as a candidate for running the ICPC World Finals contest, the following documents must be submitted.

- A text file named "License" containing the license, conforming to the the Licensing section, under which the CCS is made available to the ACM ICPC.
- A *Requirements Compliance Document* as defined below.
- A *Team Guide* as defined below.
- A *Judge's Guide* as defined below.
- A *Contest Administrator's Guide* as defined below.
- A *System Manager's Guide* as defined below.

Requirements Compliance Document

The CCS must include a *Requirements Compliance Document* in PDF format, that for each requirement (referencing by section number) in this document confirms that the CCS conforms and explains how that is. In the event that a configuration item is provided by using services of the underlying operating system rather than facilities directly implemented or controlled by the CCS itself, this fact must be explicitly stated.

Team Guide

The CCS must include a "Team Guide" in PDF format. The Team Guide must provide all the necessary instructions showing how a contest team uses the functions of the team interface.

Judge's Guide

The CCS must include a "Judge's Guide" in PDF format. The Judge's Guide must provide all the necessary instructions showing how a human contest judge uses the functions of the human judge interface.

Contest Administrator's Guide

The CCS must include a "Contest Administrator's Guide" in PDF format. The Administrator's Guide must provide all the necessary instructions showing how contest personnel use the functions of the CCS to set up and manage a contest.

System Manager's Guide

The CCS must include a "System Manager's Guide" document in PDF format. The System Manager's Guide must describe the steps required to install and start the CCS on the OS platform specified for use in the World Finals. In the event that the CCS consists of multiple modules and/or packages, the guide must contain a description of the relationship between the modules or packages, including any specific installation and/or startup steps required for each module or package.

The System Manager's Guide must provide instructions to contest personnel explaining situations (if any) where the CCS uses functions of the underlying operating system (OS) platform to meet requirements laid out in this document. For example, if the CCS relies on the use of OS account and password management to implement requirements related to contest security and credentials, or print services provided by the OS to implement requirements related to external notifications, this must be described in the System Manager's Guide.

The System Manager's Guide must explicitly list any operating system dependencies of the CCS, including but not limited to which OS platform and version are required and which optional OS packages must be installed for the CCS to function properly.

The System Manager's Guide must explicitly list any software dependencies of the CCS. For example, any tools such as Java, PERL, WebServer, Browser, database systems, etc., which are required for correct operation of the CCS must be listed in the guide, including specific version numbers of each tool which the CCS requires for its correct operation.

Appendix: File formats

All files must be encoded in UTF-8.

Input files

contest.yaml

A YAML file consisting of a mapping with the following keys:

Key	Description
name	Name of contest
short-name	Short name of contest
start-time	Date and time in ISO 8601 format (wall-clock time that the contest starts)
duration	Duration as h:mm:ss (length of contest, in contest time)
scoreboard-freeze-length	Time length before end of contest when scoreboard will be frozen, in h:mm:ss
penalty-time	Penalty minutes for rejected submission (optional, default 20)

Example

```
# Contest configuration
---
name:                       ACM-ICPC World Finals 2011
short-name:                 ICPC WF 2011
start-time:                 2011-02-04T01:23:00Z
duration:                   5:00:00
scoreboard-freeze-length: 1:00:00
penalty-time:               20
```

system.yaml

A YAML file consisting of a mapping with the following keys:

Key	Description
default-clars	Sequence of pre-defined clarification answers. The first is the default and will be pre-selected
clar-categories	Sequence of categories for clarifications.
languages	Sequence of mappings with keys as defined below

languages

A sequence of mappings with the following keys:

Key	Description
name	Name of language
compiler	Path to compiler
compiler-args	Argument list for compiler. {files} denotes where to include the file list
runner	Path to runner. Optional, relevant for interpreted languages and languages running on a VM

runner-args	Argument list for runner

A CCS may place reasonable requirements on the configuration of languages. For instance, a CCS may require that the C compiler is configured such that the output binary will be statically linked and output a binary with name 'long_weird_name'. Such requirements should be clearly specified in the CCS documentation. The CCS should not place any restrictions on optimization options used, and should in general place as few restrictions as possible on the language configuration.

Example

```
# System configuration
---
default-clars:
  - No comment, read problem statement.
  - This will be answered during the answers to questions session.

clar-categories:
  - General
  - SysOps
  - Operations

languages:
  - name: C++
    compiler: /usr/bin/g++
    compiler-args: -O2 -Wall -o a.out -static {files}

  - name: C
    compiler: /usr/bin/gcc
    compiler-args: -O2 -Wall -std=gnu99 -o a.out -static {files} -lm

  - name: Java
    compiler: /usr/bin/javac
    compiler-args: -O {files}
    runner: /usr/bin/java
    runner-args:
```

problemset.yaml

A YAML file consisting of a mapping with the following keys:

Key	Description
problems	Sequence of mappings with keys as defined below

problems

A sequence of mappings with the following keys:

Key	Description
letter	Upper case letter designating the problem

short-name	The problem identifier, used to map to the problem data
color	Color of balloon used for this problem
rgb	RGB values for balloon used for this problem (should match color above)

Example

```
# Problem set configuration
---
problems:
  - letter:     A
    short-name: apl
    color:      yellow
    rgb:        '#ffff00'

  - letter:     B
    short-name: barcodes
    color:      red
    rgb:        '#ff0000'

  - letter:     C
    short-name: biobots
    color:      green
    rgb:        '#00ff00'

  - letter:     D
    short-name: castles
    color:      blue
    rgb:        '#0000ff'

  - letter:     E
    short-name: channel
    color:      white
    rgb:        '#ffffff'
```

groups.tsv

A text file consisting of a version line and one line for each group (super regionals at the WF) that needs to be tracked separately with regards to results. Each line has tab separated fields as defined below.

The first line has the following format

Field	Description	Example	Type
1	Label	groups	fixed string (always same value)
2	Version number	1	integer

Then follow several lines with the following format (one per team group).

Field	Description	Example	Type
1	Group ID	902	integer

2	Group name	North America	string

teams.tsv

A text file consisting of a version line and one line for each team in the contest. Each line has tab separated fields as defined below.

The first line has the following format

Field	Description	Example	Type
1	Label	teams	fixed string (always same value)
2	Version number	1	integer

Then follow several lines with the following format (one per team).

Field	Description	Example	Type
1	Team Number	22	integer
2	External ID	24314	integer
3	Group ID	4	integer
4	Team name	Hoos	string
5	Institution name	University of Virginia	string
6	Institution short name	U Virginia	string
7	Country Code	USA	string ISO 3166-1 alpha-3

accounts.tsv

A text file consisting of a version line and one line for each non-team account in the contest. Each line has tab separated fields as defined below.

The first line has the following format

Field	Description	Example	Type
1	Label	accounts	fixed string (always same value)
2	Version number	1	integer

Then follow several lines with the following format (one per account).

Field	Description	Example	Type
1	Account Type	judge	string
2	Full Name	Per Austrin	string
3	Username	austrin	string
4	Password	B!5MWJiy	string

Account Types are: **team**, **judge**, **admin**, **analyst**.

For accounts of type **team** username is on the form "team-nnn" where "nnn" is the zero padded team number as given in teams.tsv.

logos.tar.gz

The file contains the logos for all teams, in a tar archive and compressed using gzip.

The archive must contain a folder named "logos", with a logo in .png format, containing an alpha channel, for each team in the contest. The logo must be named <team number>.png, where team number is padded with 0 to 4 digits. Example: 0042.png. This folder may not contain any other files. If possible, the logos should be as square as possible and 600x600 pixels or larger (but don't upsize images or add unnecessary transparent borders).

The archive may contain other files and folders. These must be ignored by the CCS when importing the logos.

Output files

scoreboard.tsv

A text file consisting of a version line and one line for each team in the contest, sorted in position order with alphabetical order on team name as tie breaker. Each line has tab separated fields as defined below.

The first line has the following format

Field	Description	Example	Type
1	Label	scoreboard	fixed string (always same value)
2	Version number	1	integer

Then follow several lines with the following format (one per team).

Field	Description	Example	Type
1	Institution name	University of Virginia	string
2	External ID	24314	integer
3	Position in contest	1	integer
4	Number of problems the team has solved	4	integer
5	Total Time	534	integer
6	Time of the last accepted submission	233	integer
6 + 2*i* - 1	Number of submissions for problem *i*	2	integer
6 + 2*i*	Time when problem *i* was solved	233	integer

The External ID for a team can be found in the teams.tsv.

The time when problem was solved must be -1 if the problem was not solved.

The time of last accepted submission must be -1 if team has not solved a problem.

results.tsv

A text file consisting of a version line and one line for each team in the contest, sorted in rank order with alphabetical order on team name as tie breaker. Each line has tab separated fields as defined below.

The first line has the following format

Field	Description	Example	Type
1	Label	results	string (the constant "results")
2	Version number	1	integer

Then follow several lines with the following format (one per team).

Field	Description	Example	Type
1	External ID	24314	integer
2	Rank in contest	1	integer or empty string
3	Award	Gold Medal	string
4	Number of problems the team has solved	4	integer
5	Total Time	534	integer
6	Time of the last submission	233	integer
7	Group Winner	North American	string

Group Winner is a string with the name of the group if the team is the group winner, otherwise empty.

The External ID for a team can be found in the teams.tsv.

If the team is not ranked (has no assigned rank) then the Rank in contest field is empty.

Award is a string with value "Gold Medal", "Silver Medal", "Bronze Medal", "Ranked" or "Honorable" as appropriate, see Scoring Data Generation for details.

submissions.tsv

A text file consisting of a single line per submission. Each line has tab separated fields as defined below.

Field	Description	Example	Type
1	submission id	7	integer
2	team number	12	integer
3	short problem name	railway	string from contest.yaml
4	submission time in ms	2033497	long integer
5	judge response acronym	AC	string

Navigation menu

Views

- History
- PDF Export

Personal tools

- Log in

Navigation

- Main page
- Recent changes
- Random page
- Help

Search

Go Search

Tools

- What links here
- Related changes
- Special pages
- Permanent link
- Page information