Contest Control System Specification Table of Contents

- 1 Introduction
  - 1.1 Overview
  - 1.2 Rationale
- 2 General Requirements
  - 2.1 Advance Configuration
  - 2.2 No outside contact
  - 2.3 On-site representation
  - 2.4 Licensing
  - 2.5 Platform
  - 2.6 Data Persistency
    - 2.6.1 Contest Configuration Persistency
    - 2.6.2 Contest State Persistency
  - 2.7 Secure Authentication
    - 2.7.1 Creation of Authentication Data
    - 2.7.2 Logging Out
    - 2.7.3 Underlying Implementation
  - 2.8 Network Security
  - 2.9 Timestamps and Ids for Runs, Clarifications and Notifications
  - 2.10 Limitations
    - 2.10.1 Number of Problems
    - 2.10.2 Number of Teams
    - 2.10.3 Test Data Files
    - 2.10.4 Other Required Specifications
- 3 Contest Configuration
  - 3.1 Importing Contest Configuration File
  - 3.2 Predefined Clarification Answers
  - 3.3 Clarification Categories
  - 3.4 Programming Languages
    - 3.4.1 Supported Languages
    - 3.4.2 Language Options
  - 3.5 Contest Problems
  - 3.6 Importing Team Registration File
  - 3.7 Configuration Change
- 4 Admin Interface
  - 4.1 Account Disabling
  - 4.2 Changes To Authentication Data
  - 4.3 Starting the Contest
  - 4.4 Adjusting for Exceptional Circumstances
    - 4.4.1 Removing time intervals
    - 4.4.2 Changing contest length
    - 4.4.3 Adding penalty time
  - 4.5 Problem States: Enabled, Paused, and Disabled
  - 4.6 Rejudging
    - 4.6.1 Automatic Rejudging
    - 4.6.2 Verifying Rejudgement Results
    - 4.6.3 Run Selection
  - 4.7 Manual Override of Judgments
  - 4.8 Scoreboard Display
  - 4.9 Freezing the Scoreboard

- 4.10 Suspension of notification of accepted runs
- 4.11 Finalizing the Contest

- 5 Team Interface
  - 5.1 Submitting a Run
    - 5.1.1 Submission Mechanism
    - 5.1.2 Submission Contents
    - 5.1.3 Team Viewing of Submission Status
    - 5.1.4 Submission Security
  - 5.2 Submitting a Clarification
    - 5.2.1 Clarification Mechanism
    - 5.2.2 Clarification Content
    - 5.2.3 Team Viewing of Clarification Status
    - 5.2.4 Clarification Security
  - 5.3 Broadcast Messages
    - 5.3.1 Team Viewing of Broadcast Messages
  - 5.4 Notifications
  - 5.5 Scoreboard Display
  - 5.6 Current Problem State
- 6 Judge Interface
  - 6.1 Simultaneous Judging
  - 6.2 Viewing Runs
  - 6.3 Handling Clarifications
  - 6.4 Issuing Broadcast Messages
  - 6.5 Scoreboard Display
- 7 Judging
  - 7.1 Automatic Invocation
  - 7.2 Prevention of Auto-judge Machine Starvation
  - 7.3 Prohibited Operations
  - 7.4 Judge Responses
  - 7.5 Assigning a Judgment
    - 7.5.1 Judging With a Single Input File
    - 7.5.2 Judging With Multiple Input Files
    - 7.5.3 Exceptional Judgments
  - 7.6 Validators
    - 7.6.1 Additional Requirements
    - 7.6.2 Default Validator
  - 7.7 Balloon Notifications
- 8 Scoring
  - 8.1 Scoring Data Generation
  - 8.2 Scoreboard
- 9 Data Export
  - 9.1 Event Feed
  - 9.2 Scoreboard Data File
  - 9.3 Final Results Data File
- 10 Testability Requirements
  - 10.1 Scriptable Submissions
- 11 Documentation Requirements
  - 11.1 Standard Compliance
  - 11.2 Team Guide
  - 11.3 Judge?s Guide
  - 11.4 Contest Administrator?s Guide
  - 11.5 System Manager?s Guide

- 12 Certification Process
  - 12.1 Items Which Must Be Submitted
- 13 Appendix: File formats
  - 13.1 contest.yaml
    - 13.1.1 languages
    - 13.1.2 problemset
    - 13.1.3 Example
  - 13.2 groups.tsv
  - 13.3 teams.tsv
  - 13.4 scoreboard.tsv
  - 13.5 results.tsv
  - 13.6 userdata.tsv

Introduction

This document specifies a standard 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 version of the standard is 1.0 published September 30th, 2011.

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 standard, 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: run 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.

Every operation or specification identified in this standard using the words "shall" or "must" is required; any system not meeting that requirement will not be considered as a candidate for running the World Finals (WF).

Use of the word "should" in regard to an operation or specification in this standard means that the operation or specification is optional, in the sense that failure to meet or provide the specification or operation will not automatically disqualify a CCS from being considered as a candidate for running the World Finals. Note however that use of the word "should" identifies an operation or specification that is *desirable* and *preferred* for running the World Finals; see Certification Process for information regarding criteria used for choosing the actual CCS to be used for running a given World Finals.

A contest control system (CCS) 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 standard. Any submitted CCS which meets all of the requirements defined in this standard 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 CCS?s which have been accepted as candidates prior to the acceptance deadline. Significant weight will be given to those accepted candidates which also implement a large number of those operations and specifications identified as optional (by the use of the word "should") in the standard.

Rationale

This standard is intended to specify functional requirements, not implementation requirements. Nothing in this standard 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.

This standard is 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 standard 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 standard.

General Requirements

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 shall not be required to make any changes once the contest starts.

No outside contact

> The CCS must be able to run at the world finals site without contact to anything outside the contest network.

On-site representation

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

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 actual use in the ICPC World Finals for at least the two years following the submission. The CCS may not require obtaining any third-party license in order to be used by the ACM-ICPC.

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 several auto-judging machines that are identical to the team machines, and 2 servers. The CCS must support a robust mechanism that allows the contest to continue with minimal disruption in case of hardware failure.

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 power 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 also support persistence of the contest state once the contest has been started. This means that following a shutdown of the CCS, or a power 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 shutdown or power failure. The contest state for purposes of this requirement includes all data regarding submitted runs, submitted clarification requests, answers to clarification requests, run 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.

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 insure 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.)

Creation of Authentication Data

If the CCS uses a login/password mechanism, it should generate login/password information for all users. It should also, for each type of user, output a complete list of username/password pairs in a file using the userdata.tsv format.

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.

Underlying Implementation

The CCS may optionally 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.

Network Security

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

Timestamps and Ids for Runs, Clarifications and Notifications

A timestamp and an integral sequence number, called an Id, are assigned to runs, clarifications, and balloon notifications. Runs and clarifications are assigned an id and a timestamp at the time they enter the system, and balloon notifications are assigned an id and a timestamp at the time they are generated.

- ◊ The id sequences are separate for runs, clarifications, and notifications i.e., there are both a run 1, a clarification 1, and a notification 1.
- ◊ Ids always increase by one.
- ◊ Run j cannot have a timestamp smaller than run (j-1). The same holds for the ids of clarifications and notifications.
- ◊ Timstamps must have granularity finer than seconds.

Limitations

Number of Problems

The CCS must not impose an upper limit on the number of different contest problems.

Number of Teams

The CCS must not impose an upper limit on the number of teams.

Test Data Files

The CCS must not impose an upper limit on the number of test data files for a problem, or their sizes.

Other Required Specifications

Here are a number of specifications that the CCS must implement.

1. Event Feed Specification
2. Problem format specification - problem load file specification
3. Output validator specification - validate/test team run output, CCS must implement a default output validator
4. Input format validator specification

Contest Configuration

Importing Contest Configuration File

The CCS must be able to read the contest configuration file (contest.yaml) containing the contest configuration and use the data in that file to configure the CCS.

Predefined Clarification Answers

The CCS must provide the ability to configure predefined answers to clarification requests (e.g., "No response, read problem statement." and "This will be answered after the practice session."), 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 should be the "default" answer.

Clarification Categories

It must be possible to configure "categories" to which clarification requests can be assigned. 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 "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 Programming 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 configured using the ICPC problem format. The CCS must have full support for this standard. All problems that are added or changed must be verified as defined in the problem standard.

Importing Team Registration File

The CCS must be able to read an ICPC Contest Management System (CMS)-generated initialization file containing a list of teams registered for the World Finals, and use the data in this file to create automatically 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 format and content of the CMS-generated initialization file is given in the teams.tsv and groups.tsv files.

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.

If a user is currently logged in when the associated account is being disabled, that user must become logged out of the system.

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 should 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.

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 wall-clock time T_0 and T_1 means that all runs received between T_0 and T_1 will have the same contest-time and that the contest-time for all runs received after T_1 will have a contest-time that is T_1-T_0 lower than if the interval was not removed.

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 amount of penalty time to be added into that team's total penalty time.

Problem States: Enabled, Paused, and Disabled

Each contest problem can be in one of three mutually exclusive states: enabled, paused, and disabled. The CCS must provide a mechanism to enable, pause and disable each contest problem separately, or all at once, without the need of stopping the contest.

When a problem is enabled:

1. Teams must be able to submit solutions to the problem.
2. Unjudged submissions to the problem must be automatically judged.
3. Submissions on the problem must be included in the scoreboard.

When a problem is paused:

1. Teams must be able to submit solutions to the problem as if it was enabled.
2. Automatic judging of the problem must be disabled.
3. Submissions on the problem must be included in the scoreboard.

When a problem is disabled:

1. Teams must not be able to submit solutions to the problem. The problem name must be removed or disabled from solution submit interfaces where present.
2. Automatic judging of the problem must be disabled.
3. Submissions on the problem must not affect the scoreboard at all, including submissions received prior to disabling.

Rejudging

Automatic Rejudging

Rejudging a set of runs must be done in the following steps:

1. A user selects a set of runs to be rejudged, as described in Run Selection below.
2. The selected runs are run and judged in the same manner as for a newly arrived run. For each selected run a new judgment is determined, *but not assigned to the run*.
3. A user *commits* the results of the rejudging, whereupon all the selected runs are assigned the judgments determined in the previous step.

If the results of a rejudge have not been committed when a new rejudge is started, the results of the old rejudge are discarded.

Verifying Rejudgement Results

When a set of runs are being rejudged as described in Automatic Rejudging, the CCS must provide a way for the user to see the changes in judgments that would result if the rejudge is committed.

Run Selection

The CCS must support the ability to rejudge a set of run submissions. This means that it must be possible to provide the CCS with a filter of runs to be rejudged, based upon which the CCS automatically selects all runs matching the filter, and automatically rejudges them.

The CCS must support these filters to be any combination of filters of the following type.

1. A specific (single) run.
2. All runs which have been submitted for a specific problem.
3. All runs which have been submitted using a specific language.
4. All runs which have been assigned any specific one of the allowable run judgments as defined in Judge Responses, or all runs that received any judgment other than "Accepted" (that is, all rejected runs).
5. All runs which have been judged 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 judge runs.

Thus, for example, it must be possible to select "all rejected runs for problem B", "all Time Limit Exceeded runs using Java for problem C", or "all runs in C++".

Manual Override of Judgments

The CCS must support the ability to assign, to a single run, an updated judgment chosen from among any of the allowed run judgments as defined in Judge Responses, or else to mark the run as deleted. Deleted runs have no effect on scoring.

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 should be updated at least every 30 seconds.

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 should 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 should be treated as pending on a frozen scoreboard.

The exact phrase displayed on the frozen scoreboard should be:

```
The scoreboard was frozen with XX minutes remaining - solutions submitted in the last XX minutes
```

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.

Suspension of notification of accepted runs

It must be possible to suspend the notification of accepted runs. When this function has been invoked, the following should happen:

1. Any run that changes to Accepted should be shown as pending in the team interface.
2. A broadcast message saying that notification of accepted runs has been suspended must be sent to all teams.

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 Event_Feed#finalized element will be sent to 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 time is not over).
2. There are unjudged runs.
3. There are runs judged as Judging System Error.
4. There are unanswered clarifications.

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.

Submitting a Run

For purposes of this standard, the terms *run* and *submission*, and any combination thereof, refer to a set of source code files submitted as a single unit at one time to the judging system by a team as a proposed solution to a given problem.

Submission Mechanism

The CCS must provide each team with the ability to submit a run to the judging system.

Submission Contents

A team must be able to specify, for each run 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 arbitrarily many files in a given submission and must allow teams to make submissions for any defined contest problem and 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 run the team has submitted, 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 insure that no team can learn anything about the run submissions of any other team (other than what can be deduced from the scoreboard).

Submitting a Clarification

For purposes of this standard, the terms *clarification*, *clarification request*, and *clar* all refer to a message sent from a team to the judges asking for clarification regarding a contest problem.

Clarification Mechanism

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

Clarification Content

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

Team Viewing of Clarification 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 if any response has occurred yet.

Clarification Security

The CCS must insure that no team can see the clarification requests of any other team, except as provided in the section Human 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 should be updated at least every 30 seconds.

Current Problem State

The CCS must provide a mechanism for teams to see the current state (enabled, paused, or disabled) of each contest problem.

Judge Interface

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

Simultaneous Judging

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

Viewing Runs

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

1. See a list of all submitted runs, where the list includes (for each run) the contest-relative time at which the run was submitted, the problem for which the run was submitted, the language specified in the run submission, and any judgments applied.
2. Sort the list of runs by submission time (newest runs first).
3. Filter the list of runs by:
    1. Problem
    2. Team
    3. Language
    4. Judgement applied.
4. View and download the output produced by the program when run against the specified input data
5. View and download the source code contained in any specific submitted run
6. View the compiler output resulting from compiling any specific run utilizing 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 submitted run
8. View and download the judge's input data file associated with the contest problem for any specific submitted run
9. View and download the "judge's output" (the "correct answer" file) associated with the contest problem for any specific submitted run
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

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-relative 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 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 category of a clarification.

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 should be updated at least every 30 seconds.

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

Judging

Automatic Invocation

The CCS must provide the ability to invoke an automated judging mechanism when a team submits a run. This mechanism must automatically (that is, without human intervention) detect each newly submitted run, and for each such run must automatically:

1. Compile (if appropriate for the language) the program contained in the submitted run, enforcing the compilation time limit.
2. Execute the program contained in the submitted run, 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 by terminating the execution of the submitted program if it exceeds any of these limits.

- 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 submission to use a *wall time* that is at most twice the execution time limit (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 standard as a validator, passing to it the output generated by the program specified in the submitted run and getting back from it an indication of what judgment is to be applied to the run (see the External Validators section).
6. Assign an appropriate judgment to the run.

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

Prohibited Operations

The CCS must automatically prevent a submitted program from performing prohibited operations.

The prohibited operations are:

1. Using libraries except those explicitly allowed
2. Executing other programs
3. Creating new processes
4. Creating new threads
5. Reading any files
6. Creating files
7. Sending signals to other programs
8. Side-stepping time or memory limits
9. Sending or receiving network traffic

Judge Responses

The CCS must answer each submitted run with one of the following responses:

◊ **Compile Error**
◊ **Run-Time Error**
◊ **Time Limit Exceeded**
◊ **Wrong Answer**
◊ **Accepted**
◊ **Security Violation**
◊ **Judging System Error**

When manually overriding the judgement the following could also be used:

◊ **Deleted**

Assigning a Judgment

The next two sections define how to assign a judgment to a program for problems with a single input file and with multiple input files, respectively. Note however that the **Judging System 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 System 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** must be the judgment.

Validators

The CCS must support problem-specific Output validators as described in the problem format standard. As described there, the output validators must adhere to the output validator standard.

The CCS will execute the Team's program and then execute the Output validator.

Additional Requirements

A CCS supporting this standard must safeguard against faulty validators. For instance, if a validator were to produce excessively large feedback ?les, or crash, the CCS should 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 ?les of their own choosing.

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

Default Validator

The CCS must provide a default "diff" capable of handling the options specified in the problem format standard.

Balloon Notifications

The CCS must have an ability to send a message, called a balloon notification, for every run which have received an accepted judgment. It also must send revocations or updates resulting from rejudging.

The notification includes the following

1. A sequence number of the notification.
2. Notification type: new / revocation / update
3. Identification of the team that submitted the run.
4. The contest time at which the run was submitted.
5. A timestamp
6. The problem to which the judgment applies.
7. The color of balloon which applies to that problem (see Configuring Problems, below).
8. A list of all problems solved by the team so far, along with their balloon colors.
9. "First in contest" if this is the earliest submitted run that is accepted.
10. "First for problem" if this is the earliest submitted run that is accepted for this problem.
11. "First for team" if this is first notification of acceptance for this team.

As submission ids increase with time, the notion of "first" is based on the id of a submission. A revocation can occur when a previously accepted run is rejected due to rejudging. Similarly, rejudging of one run can change whether another run was "first" to solve a problem or not, requiring an update. If a previously revoked balloon should be awarded, this also should result in an update notification.

Note that run judgements may not come in the same order as submissions, meaning that the first notification of acceptance does not have to be for the earliest submitted accepted run.

The format of balloon notifications is defined in Event Feed Notifications.

The sequence numbers assigned to the external notifications are used to detect if a notification is missing (e.g., due to printer malfunction). The CCS must provide at least one of the two following features:

1. the CCS must be able to resend a previously sent message with a given sequence number upon request

2. the CCS must have an interface listing all notifications, and allowing a user to access the data included in any one of them. Such a complete list of notifications must be protected in such a way that teams cannot access it.

It is optional whether the CCS also provides the ability to send notifications for runs which receive rejected judgments, but it must be possible to configure the CCS to only send notifications for accepted runs.

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 20 minutes for each previous rejected run 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 by number of problems solved (descending), total penalty time (ascending), last accepted submission (ascending).
7. The *rank* of a team is the 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 System Error**. Pending judgements have no effect on scoring.

Runs marked as **Deleted** have no effect on scoring.

Scoreboard

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

The scoreboard should include at least the following information.

For each team:

1. university name and logo, country flag,
2. team position,
3. number of problems solved,
4. total penalty time.

For each team and problem:

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

Testability Requirements

Scriptable Submissions

The CCS must provide a mechanism by which team submissions of both runs and clarification requests can be scripted for automated testing. In other words, the CCS must provide a command-line based interface allowing an external program to submit a run to the CCS and to submit a clarification request to the CCS.

The run submission and clarification submission command-line interfaces must provide mechanisms for specifying all parameters that would be specified by a team utilizing the corresponding interactive interface, including that the command-line interfaces must accept and perform validation of user credentials before accepting a run or a clarification request.

The requirements of this subsection are intended to provide a method by which the World Finals Director of Operations can perform automated testing on the CCS during the Certification Process (see Certification Process). The CCS implementers should assume that the scripting interfaces will be invoked multiple times in rapid succession, in arbitrary order with arbitrary parameters, by external scripts; care should therefore be taken by the implementers to avoid any design or implementation characteristics which limit or prohibit the ability of an external Test Harness framework to operate in this manner.

Documentation Requirements

Standard Compliance

The CCS must include a *Standard Compliance* document in PDF format, that for each requirement (referencing by requirement number) in this standard document confirms that the CCS conforms and explains how the functionality is implemented.

Team Guide

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

The Team Guide must also describe the way in which judge?s responses to submitted runs and clarifications are received by teams.

Judge?s Guide

The CCS must include a ?Judge?s Guide? document 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? document 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.

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, the administrator?s Guide must explicitly state this fact, and reference the System Manager?s Guide (see below) as the source of information on handling that portion of configuration and preparation for the 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 standard. 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.

The System Manager?s Guide must describe the steps required to use each of the ?Testability Interfaces? described in the section on Testability. For example, the guide must describe how to invoke the command-line tool which supports submitting runs from a team, as well as describing the use of all other required testability interfaces.

Certification Process

Items Which Must Be Submitted

In order for a given CCS to be considered as a candidate for running the ICPC World Finals contest, a single archive file (e.g. .zip, .tar, .tar.gz, .tgz, etc.) must be submitted to the World Finals Director of Operations. The archive file must contain all of the following items:

1. The complete set of files, modules, or packages making up the executable components of the CCS;
2. PDF?s for each of the documents listed under Documentation Requirements;
3. A short ASCII text file named ?License? containing the license, conforming to the the License section, under which the CCS is made available to the ACM ICPC; and
4. A short ASCII text file named ?ReadMe? containing instructions on how to properly unpack the archive file to set up the installation process.

Appendix: File formats

All files should be encoded in utf-8.

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	Time when scoreboard will be frozen in contest time as h:mm:ss
default-clars	Sequence of pre-defined clarification answers. The first will be pre-selected
clar-categories	Sequence of categories for clarifications.
languages	Sequence of mappings with keys as defined below
problemset	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.

problemset

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

```
# Contest configuration
---
name:              ACM-ICPC World Finals 2011
short-name:        ICPC WF 2011
```

```
start-time:        2011-02-04 01:23Z
duration:          5:00:00
scoreboard-freeze: 4:00:00

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:
  - 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	Reservation 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	USA	string ISO 3166-1 alpha-3

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	Reservation 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 Reservation 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	Reservation ID	24314	integer
2	Rank in contest	1	integer
3	Award	gold	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	Region Winner	North America	string

Region Winner is a string with the name of the region if the team is the region winner, otherwise empty.

The Reservation 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", "silver", "bronze", "ranked" or "honorable" as appropriate, see Scoring Data Generation for details.

userdata.tsv

A text file created by a CCS consisting of a version line and one line of information for each account in the contest.

The first line has the following format

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

Field	Description	Example	Type
1	Account Type	team	string
2	Account Number	42	integer
3	Full Name	University of Virginia	string
4	Username/login	team-001	string
5	Password	B!5MWJiy	string

Account Types include: team, judge, admin, analyst, etc.

Account numbers for teams must be the same as the corresponding team numbers in teams.tsv.