User:Douglas Lane

From ICPC-Contest Control Standard
Jump to navigationJump to search

Douglas A. Lane

Sacramento California
ICPC SysOps Team
CSUS Programming Contest Development and Support Team
laned@ecs.csus.edu

□

Contents

2015 for 2016 Standard to do list

1. Add contest.tsv as input file to CCS Spec (add above groups.tsv), format at: http://pc2.ecs.csus.edu/wiki/Contest.tsv This file contains some information that is in contest.yaml.

2. Merge/add http://pc2.ecs.csus.edu/wiki/Contest_Data_Package#Input_Files
3. (Partial done) Add "Draft" comment on each CDP wiki page. Something like at the top of the article: *This is a draft standard for the ICPC 2016 Finals, upon approval this draft status will change to Final Approved status.*
4. Change Problem Format link to link to new Problem Format article https://clics.ecs.baylor.edu/index.php/Main_Page
5. images vs image, teams vs team directories
6. png vs jpg images

1. ~~Replace contents of Problem Format with Problem_Format_2012 as a starting version.~~
2. ~~Replace any references to Kattis problem format to the wiki Problem Format in CCS section Contest Problems~~
3. ~~Remove diagram from CDP page, new diagram when CDP structure finalized, maybe~~

Old Proposed Changes/Additions

There are two additions proposed:

- Add Data file export program
- add runs.tsv
- add standings.json
- ~~add clars in EF~~
- ~~add scoreboard freeze time in contest.yaml~~
- Run Submission Interface - add support for clars

Add Data file export program

Data file export

The CCS must provide a mechanism by which data files can be extracted for automated testing. The CCS must provide a command-line based interface allowing a program to extract files.

The requirements of this extraction 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.

Output from the extractor is written to stdout.

Example usage

Note that "`extractor`" is used in this example, the requirement is to have one program the program name. The CCS Development Team can choose the name. **The name of the extract program will be documented in the** Contest Administrator's Guide.

```
$ extractor runs > runs.tsv
```

```
$ extractor results > out_results.tsv
```

command line parameters

At a minimum the run submission command line interface must accept one parameter, 'name'.

name	Appendix File format/content
results	results.tsv
scoreboard	scoreboard.tsv
runs	runs.tsv

optional command line options

These are optional command line parameters

```
-u <team id>
-w <team password>
```

optional command line parameter

A possible addition is to specify an output filename as a parameter.

```
extractor results results3.tsv
```

That would be identical to this command

```
extractor results > results3.tsv
```

New TSV files

Where: new section after 13.8 accounts.tsv

clarification.tsv

A text file consisting of a single line per clarification. Each line has tab separated fields as defined below. Special characters will be encoded to ensure that all each clarification and answer is on a single line. XML encoding will be used, ex. newline will be tabs will be &#8; etcetera.

Field	Description	Example	Type
1	clarification id	7	integer
2	submission time in ms	2033497	long integer
3	team number	12	integer
4	area	Operations	categories as defined in CCS
5	short problem name	railway	string from contest.yaml
6	Question	When is lunch served?&10#;&#8; We are very hungry.	string
7	Answer	In the lunchroom Lunch will be announced over the PA	string

Other Notes

To do items

2013 SysOps Debrief Items

1. add runs.tsv
2. add standings.json
3. add clars in EF
4. add scoreboard freeze time in contest.yaml

Additional CCS Items

- Add specification for language compiler and program execution command lines in contest.yaml
- Add clarification.tsv
- Create a CDI tool to convert the JSON into XML, and the XML into HTML via XSL

1. Source for CDI data? CCS? CDI system? - How will these be supplied/accessed?
    1. balloon colors
    2. team pictures
    3. team name (full)
    4. team name (short)
    5. university logos

Other To do items

1. See List of issues
2. Complete and post logging messages
3. groups.tsv **Comment:** Change group name from noun to adjective, e.g. South America to South American, Europe to European, from MG Douglas Lane 02:50, 31 May 2011 (CEST)
4. results.tsv **Comment:** Add Group Id or Group name or both? To allow potential ranking in regions, etc. Douglas Lane 02:52, 31 May 2011 (CEST)
5. Add event-feed-port to contest.yaml.
6. Embed file into XML, is there a standard for this?
7. Add somewhere: Textfiles should be UTF-8 encoded.
8. References to CMS, remove them data is data
9. Run Submission Interface - there is no support for clars

Follow up requirement specification

1. contest.yaml - error if there are duplicate problem letters
2. contest.yaml - warning if there are missing letters, ex. A, B, D, E (missing C)

CMS Changes

1. In Contest_Control_System#results.tsv rank in contest field is empty if team not assigned rank.
2. in .tsv files - change "file_version" to file basename

Notes

1. Upon time interval removal, all runs with changes contest-time must be re-sent to the Event Feed

Nice to haves

1. Judge sends a clarification to a specific team/user
2. Judge sends a clarification to a list of users (by site, pick from list, etc.)
3. Report or list of removed time intervals
4. Report of list of all runs affected by removed time intervals

Logging Items

1. Configuration Changes
2. Run status/info changes
    1. Remove time interval
3. admin changes run (judgment, contest-time, etc.)
4. Log when team's score is adjusted by Admin.
5. When new validator installed
6. when validator command line changes
7. when new judge's data or answer file loaded
8. when team disabled (no longer can login)
9. when problem changes state

TBD Send out a note with this example
A suggestion is that all of these events should be in the event feed directly. For example if a problem is disabled then enabled there would be two Event Feed events. If a user logs in and gets an event feed, the problem is enabled would the CCS have to store when the problem was enabled and disabled and send those events, in order, on the Event Feed? Or should the Event Feed simply give the problem as enabled ?

Problem Format

There are a number of optional in Comments in the Configuration file format, there should be examples or state that user defined values are allowed.

Event Feed

1. The CCS will provide Event Feed data using a socket connection (port).
2. The CCS will require a login/authentication before providing the Event Feed data.
3. The CCS will send out all events from the start of the contest then continue send events.
4. Any change to the run data will cause a new run event (judgement/state change)

Finalize Data

The following data must be entered before a contest can be finalized.

1. last gold medal rank
2. last silver medal rank
3. last bronze medal rank
4. number of bronze medals awarded
5. a Finalize comment

Finals Results Data (2013)

CCS Team Member gives the following files to Deputy Executive Director

1. results.tsv
2. Final Results HTML (results.html)
3. Full Results HTML (final_standings.html)

Source: 2013.1 2013 ICPC World Finals –– End of Contest Procedure (PDF).

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.

CLI interface

At a minimum the command line interface must implement the following switches:

```
-p <problem_id>

-u <user_id>

-t <unix timestamp for submission>

-f <folder containing the files in the submission. The folder must contain
all files for the submission, and no other files.>

-p, -u and -f are mandatory for all submissions. -t is optional. File names
must be identical to those used in the submission. The CCS may implement
additional switches.
```

Run Forwarding Interface

A CCS must provide a dynamic run forwarding interface (Run Forwarding Feed) through which an external program can obtain in real time a copy of each run submitted by a team. In addition, a CCS must be capable of connecting to an external program which implements a run forwarding interface and accepting in real time the run submissions provided by that run forwarding interface.

The run forwarding interface (Run Forwarding Feed) of a CCS must function according to the following specifications:

1. The CCS must listen, on a port specified in the appropriate entry in the contest.yaml file, for a connection from an external program to its run forwarding interface.
2. Whenever a connection is made on the run forwarding interface port, the CCS must transmit out on the port all of the data comprising each run submitted by a team as the contest progresses. The transmitted data must be in the same form as that required by this Standard for compliance with a CCS command-line submission interface.
3. The CCS must be capable of accepting multiple independent connections on the run forwarding interface, and must treat each connection in the same way by sending all runs submitted by a team to each connected external program.

A CCS must, at startup time, examine the contest.yaml file for a specification of a remote machine (IP address and port) to which it should connect to receive forwarded runs, and if such a specification is present the CCS must open a connection to the specified run forwarding interface. The CCS must accept each run received from the remote run forwarding interface, and must process each received run in exactly the same manner as if the run had been submitted by a team through the CCS's command-line run submission interface. It is allowable to assume that no two programs implementing a run forwarding interface (that is, listening for run forwarding connections) are running on the same machine (i.e., sharing the same port address space at the same IP address).

Run Forwarding Interface

A CCS must provide a dynamic run forwarding interface (Run Forwarding Feed) through which an external program can obtain in real time a copy of each run submitted by a team. In addition, a CCS must be capable of connecting to an external program which implements a run forwarding interface and accepting in real time the run submissions provided by that run forwarding interface.

The run forwarding interface (Run Forwarding Feed) of a CCS must function according to the following specifications:

1. The CCS must listen, on a port specified in the appropriate entry in the contest.yaml file, for a connection from an external program to its run forwarding interface.
2. Whenever a connection is made on the run forwarding interface port, the CCS must transmit out on the port all of the data comprising each run submitted by a team as the contest progresses. The transmitted data must be in the same form as that required by this Standard

for compliance with a CCS command-line submission interface.

3. The CCS must be capable of accepting multiple independent connections on the run forwarding interface, and must treat each connection in the same way by sending all runs submitted by a team to each connected external program.

   A CCS must, at startup time, examine the contest.yaml file for a specification of a remote machine (IP address and port) to which it should connect to receive forwarded runs, and if such a specification is present the CCS must open a connection to the specified run forwarding interface. The CCS must accept each run received from the remote run forwarding interface, and must process each received run in exactly the same manner as if the run had been submitted by a team through the CCS's command-line run submission interface. It is allowable to assume that no two programs implementing a run forwarding interface (that is, listening for run forwarding connections) are running on the same machine (i.e., sharing the same port address space at the same IP address).

Navigation menu

Search

Go Search

Tools

- What links here
- Related changes
- User contributions
- Logs
- View user groups
- Special pages
- Permanent link
- Page information