The ICPC Contest Data Server (CDS)

{width=50}

An ICPC Tool

Introduction

The Contest Data Server (CDS) is a software component designed to allow secure, authenticated HTTP connections to contest data residing on a contest network. It provides REST entry points for accessing a variety of contest-related services, allowing clients to access those services via standard HTTP requests.

The CDS provides authentication services using configuration data supplied by the Contest Administrator (CA). Authenticated users can be assigned *roles* by the CA; user access to specific services is governed by the role(s) assigned to them.

The set of services made available by the CDS is dependent on the facilities available on the contest network. Examples of services which the CDS knows how to support (assuming that the contest network provides the back-end data required and that the CA configures their use into the CDS) include

- an overview description of the contest (title, date, etc.)
- a Countdown Clock for the start of the contest
- the current contest time (time remaining in the contest)
- the contest configuration, as defined by any of several standard file types (contest.yaml, problemset.yaml, teams.tsv, etc.)
- the current backup copy of any specified team's home directory
- the event feed as provided by the Contest Control System (CCS)
- images for each team and for each team's University Logo
- "reaction videos" showing a team's web camera at the moment they received a run submission response from the judges
- the current contest scoreboard
- an RSS feed for the contest
- the set of files which a team submitted to the judges for a specific run

Many of these services are described in more detail in the following sections of this document; additional information regarding CDS services can be found on the [CLI Contest Data Package specification]

Note: many of the data access functions provided by the CDS require the use of a compatible *Contest Control System* (CCS) for running the contest. See the *Competitive Learning Institute*'s Contest Control System specification for details. See the documentation for the CCS being used by your contest to determine whether it meets the CLI CCS specification.

Using the CDS

Overview

The CDS is a Java Enterprise Edition (JavaEE) web application; an instance of the CDS operates by being deployed on a JavaEE application server. The ICPC Tools distribution of the CDS comes pre-packaged running on Open Liberty; the entire CDS can be run using the embedded Liberty server without any other tools.

Alternatively, the CDS can be deployed on other Application Servers such as Apache Tomcat, JBoss (now WildFly, or Jetty. However, the mechanisms and requirements for deploying web applications to application servers differ between servers. This guide does not attempt to provide tutorial details for such alternative deployments; see the Appendices and the documentation for your particular application server for additional information.

Installing the CDS

To install the CDS, download a copy of the latest version from the ICPCTools website and unzip it into any convenient directory. That's it!

Configuring the CDS

The unzipped CDS contains a single folder named *wlp* (Liberty). Among the subfolders therein you will find both *bin* and *usr* folders. The *bin* folder contains commands (see below) which are used to start and control the server and the applications it contains (in this case, the CDS). The *wlp/usr/servers/cds* folder holds the "WAR" file which contains the CDS executable code (under the *apps* folder), and also holds the *configuration files* which manage the CDS configuration.

The CDS is configured via a combination of two things: the JavaEE application server-specific configuration mechanism (for ports, virtual hosts, users, etc.), and XML configuration files for CDS settings (such as which Contest Control System (CCS) to connect to, user account passwords, etc.).

Configuring the WLP JavaEE Server The Liberty configuration is managed by a file named *server.xml* located in the *usr/servers/cds* folder. The *server.xml* file contains five important sections:

- a specification of the *HTTP and HTTPS ports* used by the server.
  - The ports associated with HTTP and HTTPS requests can changed by editing the *httpEndpoint* entry in the *server.xml* file; the defaults are 8080 and 8443 respectively.
- a list of the *security roles* and *user names* associated with those roles.
  - The security roles and associated user names can likewise be edited (but be aware that some ICPCTools assume that certain users are associated with certain roles). See below for further information about user roles.
- the location of the *users.xml* file which defines the passwords for user accounts.
  - The passwords assigned to specified user names can be changed by editing the *users.xml* file.
- a JNDI (Java Naming and Directory Interface) entry giving the path from which to load the CDS configuration details.
  - The *server.xml* configuration file must contain a single JNDI entry named "icpc.cds.config" that gives the path to load the CDS configuration and other files required by the CDS. The CDS comes preconfigured with a JNDI entry pointing to the current folder, which contains the file *cdsConfig.xml* (see below).
- a list of *features* used by the JavaEE server (not normally changed by CDS users).

Configuring CDS Accounts Access to the CDS is configured via an *accounts.yaml* (or *accounts.json*) file in the same folder as cdsConfig.xml. This account file follows the Contest API and Contest Archive Format specifications and allows you to configure any number of users to access the CDS. In addition to the spec-defined account types of admin, staff, analyst, team, the CDS supports three additional account types: spectator (receives additional non-public data like commentary), balloon (typically used for printing balloons at the World Finals) and presAdmin (users who can control presentations).

Configuring CDS Services In addition to the general configuration items described above, the CDS must be made aware what specific services are available on the contest network as well as how it should access those services in order to expose them to clients. This configuration is done via an XML file named *cdsConfig.xml*, located in the *usr/servers/cds* folder. The *cdsConfig.xml* file contains a root element *cds* which in turn contains a separate "service-defining" XML element for each service which the CDS knows how to provide. All nested elements and attributes are optional except for the *contest location* attribute, and unknown elements or attributes are ignored - this allows you to easily rename or comment-out an element to add or remove a service. If you save changes to *cdsConfig.xml* while the CDS is running the changes will be automatically applied after a few seconds.

A simple example of a *cdsConfig.xml* file is shown below.

```
  <!-- specify the location of the folder holding the contest data package
       (used to find config files, resources, etc.) -->
  <contest location="/Users/me/contests/test" recordReactions="true">
    <!-- Configure communication with the CCS -->
    <ccs url="http://myccs/api/contests/finals" user="admin" password="adm1n"/>
  </contest>
</cds>
```

The above file specifies two "service elements": *contest*, with its (required) *location* attribute which points to the location of the Contest Data Package (see below), and *ccs*, whose attributes describe how to connect to the Contest Control System. (See below for further description of the attribute values in the sample file.)

CDS Configuration Service Elements The CDS supports a variety of service-defining XML elements in the *cdsConfig.xml* file. The supported service-defining elements and their structure and functions are described below.

contest Element

```
<contest id="id" location="path" recordReactions="false"/>
```

The attributes associated with this element are as follows:

- id: an optional *id* specifying the contest id to expose, which will override any contest id used by the underlying CCS or event feed.

  If unspecified, the id will be picked up from the CCS' Contest API endpoint (e.g. 'finals' in 'http://api/contests/finals') or by the last segment of the location (e.g. 'test' in 'C:\icpc\test').

- location: specifies the full path to a contest data package (CDP) defining the organization of the contest (config files, logos, etc.). See the CLI CDP specification for details on how to organize a Contest Data Package. Note that *the location attribute is required*; the CDS will not operate if it does not have a folder to store contest related data, even when it is empty and getting all data from the CCS.

- recordReactions: an optional string ("true" or "false" [default]) instructing the CDS whether or not to record reaction videos and make them available through the contest API.

  If the value of this attribute is "true", then clients will be able to access reaction videos (webcam snippets following the moment a team sent a submission to the judges). The value of the *recordReactions* attribute should only be set "true" if the *webcam* URL template has been set and team machines are actively streaming webcam video.

- The optional *hidden* attribute specifies that a contest should be hidden from contest listings unless the request is coming from an admin or blue user. This allows contest administrators to configure the next contest prior to broadcasting it's existence. Note that this is just meant to allow some configuration to happen in private, not a way to completely secure these contests.

ccs Child Element

```
<ccs
    url=""
    user=""
    password=""/>
```

The optional *ccs* element is used to configure the CDS's access to the Contest Control System. The attributes associated with this element are as follows:

- url: the URL to a REST-based CCS
- user: the user account name to log into the CCS
- password: the password for the CCS user account

If the *ccs* element is present then the CDS will use the CCS as the master source of all contest data. The local *contest* folder is used as an initial source, cache, and can provide additional data (e.g. logos if the CCS does not provide those) but the event feed and all other contest data will come from or be overridden by the CCS.

video Child Element

```
<video
    id="43"
    desktop="http://..."
    desktopMode="lazy"
    webcam="http://..."
    webcamMode="eager"
    audio="http://..."
    audioMode="eager"/>
```

The *video* element is used if your contest supports team desktop (screen), webcam, and/or audio streaming, in which case the CDS can make these streams available to clients. Also, if the contest does support team webcam streaming, then the CDS can be instructed to record "reaction videos", which are snippets of webcam video started at the moment a team submits a run to be judged (and should the effect of capturing the team's "reaction" when they receive the judgment).

The attributes associated with the *video* element are as follows:

- id: an optional team id. If this is specified, the video URLs below will be configured only for the given team. If not specified then the video URLs will be configured for all teams.
- desktop: an optional template for the URL at which the CDS should access team desktop streams. The URL template may include two substitution variables:
  - "*{0}*", which will be replaced with the team id by the CDS.
  - "*{host}*", which will be replaced by one or more host entries (see host configuration element for more details).

  For example, if each team machine was on a single /24 subnet (i.e., the first three octets of the team machine IP addresses were all the same) and further the last octet of each team machine's IP address was the team number, then a *desktop* attribute of the form *http://a.b.c.{0}:9090* would direct the CDS to access each team's desktop at port 9090 on the machine with IP address a.b.c.X, where 'X' was the team id (typically team number). As another example, *webcam* URL templates of the form *http://host/{0}* or *http://host:{0}* would direct the CDS to access each team's webcam at a different web-based context root or port, respectively. Note that the URL template mechanism works the same

for the *desktop*, *webcam*, and *audio* attributes. In any case, it is the responsibility of the Contest Administrator to ensure that each team machine is actively outputting desktop video, webcam video, and/or audio streams at the specified URL. (See the Appendices for information on setting up video streaming for team machines.) In order to support easy testing, the desktop attribute may also point to a local file instead of a URL, e.g. "c:/desktop_sample.m2ts". Desktop and webcam files for testing should be roughly 1 min in length as they're streamed every 60s.

- desktopMode: an optional string that tells the CDS how to treat the desktop stream. See below for details.
- webcam: an optional template for the URL at which the CDS should access team webcam streams. The URL template may include the substitution *{0}* or *{host}*, which will be replaced with the team id by the CDS. See the desktop attribute for further details.
- webcamMode: an optional string that tells the CDS how to treat the webcam stream. See below for details.
- audio: an optional template for the URL at which the CDS should access team audio streams. The URL template may include the substitution *{0}* or *{host}*, which will be replaced with the team id by the CDS. See the desktop attribute for further details.
- audioMode: an optional string that tells the CDS how to treat the audio stream. See below for details.

Streaming Modes

The CDS video and audio streams support 4 modes which are described below. Aside from the "direct" mode, all other modes will pass the streams through the CDS. The CDS is a video/audio aggregator capable of multiplexing to many clients - i.e. regardless of how many clients are connected to the same team stream, there is ever only one connection to the team machine. The CDS can also be dual-homed (on two networks), allowing clients that can't directly access teams to see their video streams.

Your mileage may vary and should be tested in advance, but CDS streaming tends to be network limited. i.e. an average laptop or desktop machine running the CDS can stream up to 300 HD webcams at a time (~3Mb/HD stream), which at 900Mb/s saturates a 1Gb ethernet connection.

Depending on how the team machine is configured to do streaming, there can be an initial connection delay of several seconds. To help with this the CDS has the "lazy close" and "eager" modes described below.

These are the supported modes:

Mode	Description
direct	Exposes the stream URL directly to clients in the feed. The CDS will not access the URL unless you enable th
lazy	Gives clients a URL to the CDS. The CDS will connect to the source only when the first client requests a part
lazy_close	Same as lazy, but the CDS will stay connected after the last client drops. This will allow subsequent calls to c
eager	Cause the CDS to connect to all streams immediately, and stay connected even after all clients disconnect - us

test Child Element

```
<test
    countdown=""
    startTime=""
    multiplier=""/>
```

The *test* element is used to enable test (*contest playback*) mode. When you enable test mode the CDS will attempt to play back a past contest based on the content of the specified *contest location* (see the *contest* element, above) -- meaning, the CDS will provide responses to service requests as if the requested data came from the specified contest, at the time rate and sequence order in which they actually occurred in that contest. The attributes associated with this element are as follows:

- countdown: a string giving the number of seconds before the contest playback should start
- startTime: a specific start time, e.g. "2018-02-19T12:30:00.000-05"
- multiplier: a string giving a numeric multiplication factor for contest time; for example, "50.0" will playback a 5-hour (300-minute) contest in 6 minutes.

Only one of the countdown and startTime attributes can be set at a time! Several services normally require additional hardware or software to run; e.g. /video/desktop requires a team machine that's capturing and streaming desktop video. To avoid this requirement during a test (contest playback), the CDS allows you to put into the contest location (CDP) a *test* folder that contains some sample data. The following is a list of the sample files that are supported:

- reaction.m2ts
- submission.json
- backup.tar.gz

video Element

```
<video
    name="ICPC Live"
    url="http://..."
    mode="lazy"/>
```

The global *video* element is used if you want to add additional video streams into the channel mixer.

The attributes associated with this *video* element are as follows:

- name: an name to represent this video stream
- url: a video URL at which the CDS can access a video stream
- mode: an optional string ("lazy", "lazy close", or "eager") that tells the CDS how to access the desktop stream. "Lazy" will connect only when the first client requests the stream and disconnect when the last client drops. "Lazy close" will only connect when a client requests the stream but will stay connected afterward, thus allowing any subsequent clients to connect faster. "Eager" will connect immediately and stay connected even when there are no clients - using up lots of resources, but allowing all clients to connect as fast as possible.

domain Element

```
<domain>
  <user name="presAdmin"/>
  <user name="presentation1"/>
  <user name="presentation2"/>
</domain>
```

The global *domain* element is used if you have multiple presentation administrators and clients and want to set up separate administrative domains. You can define multiple domains. Each domain should have at least one admin or presentation admin user, and at least one non-admin user that will be used to display presentations.

By defining a domain you are restricting a set of administrators who may control a given set of user's presentations. For example, the domain above means that only presAdmin may control presentation1 and presentation2 - other administrators will not see these clients unless there are other domains defined. For more complex scenarios domains can overlap in any way, e.g. the primary admin may be included in more than one domain or presentation users may be in multiple domains to allow more than one admin to control them.

The attribute associated with the *user* element is as follows:

- name: the name of a user, which must match one of the existing users.

host Element

```
<host host="10.0.0.{0}"/>
```

The global *host* element allows you to map hostnames to teams, either for ease of configuring streaming video urls or auto-login. Each element uses a substitution variable {0} (the team id) to map every team to the given host.

The attributes associated with the *host* element is as follows:

- host: an IP address or host name.

Starting the CDS

The CDS distribution includes a folder *bin* (under the *wlp* folder) that contains a variety of scripts and other tools for performing CDS-related functions. Among these scripts are a Bourne shell script named *server*, and a corresponding Windows batch file named *server.bat*. These are the scripts that are used to start the WLP application server on the corresponding platform. (Note that this structure allows the same command, "*server*", to be used regardless of the platform.)

The "server" script requires one or more parameters telling the WLP application server what to do. In particular, the following table shows the most common commands for running the CDS on the WLP Application Server (the listed commands assume the user is in the *wlp* folder and are shown in their Linux form; replace the "/" characters with "" on a Windows machine).

Command	Effect
/bin/server start cds	Start the CDS in background mode (allows running other commands from the same prompt, but d
/bin/server stop cds	Stop the CDS
/bin/server run cds	Start the CDS in foreground mode (ties up the console window, but allows display of server output
/bin/server status cds	Displays the current status of the CDS
/bin/server list	List the servers which WLP knows about

Accessing CDS Services

Once the CDS is configured and started, it provides REST entry points (URLs) which clients can access to obtain services. The full form of a URL is always

```
<protocol>://<host>:<port>/<contextRoot>/<pathWithinApp>
```

The CDS defaults to using HTTP and HTTPS protocols at ports 8080 and 8443 respectively (although the port is configurable, as described earlier), and the default *contextRoot* for the CDS (that is, the place where the application server expects to find application pages) is "/", so a typical URL for accessing CDS services is

```
https://<host>:8443/<serviceName>
```

where *host* is the IP address of (or a hostname which resoves to the IP address of) your CDS machine, and *serviceName* is the name of one of the services provided by the CDS.

For example, assuming the CDS is running on a machine with hostname "*cds*" under the application server's "/" context root (the default), a client could access the "overview" service by referencing the following URL:

https://cds/overview

Similarly, accessing

https://cds/images/team/3

would return the picture for team #3, and accessing

http://cds/video/camera/17

would connect to the video camera stream for team #17. (Note that all of the above examples presume that the corresponding service has been configured into the CDS, as described earlier.)

A complete list of the services provided by the CDS can be found on the Contest Data Server web page.

Updating ICPC Clients

When using any of the ICPC Tools clients (presentations, balloon printer, coach view, and resolver) with the CDS, they will check the CDS for newer versions on startup, and will automatically download and replace themselves if a newer version is available. To use this function, download the zip file of a newer version of a tool and place it in the CDS's config/present folder.

For example, if a balloon client had the 2.2.87 version installed and you downloaded the 2.2.90 version and placed it at wlp/usr/servers/cds/config/present/balloonUtil-2.2.90.zip, the next time the balloon client is started or restarted it will update to 2.2.90. The presentation admin can be used to force a restart of any client.

Appendix

Setting up Video Streaming on Team machines

In order for the CDS to be able to respond to video service requests (for example, access to team machine desktop or team webcams), two things must be done. First, the CDS must be configured to look for team desktop and/or webcam feeds (see the description of the *video* service element in the *cdsConfig.xml* file, above). Second, the Contest Administrator must arrange for team machines to generate appropriate video streams on the URLs which are specified in the *video* element.

Any mechanism can be used to generate the video streams on the team machines. At the World Finals, the free open-source package called VLC is used. Specifically, each team machine has VLC installed, and two instances of VLC are running on each team machine: one to generate the team desktop stream, and a second to generate the team webcam stream. The commands used to generate these streams are shown below.

Note that while this works at the World Finals, it is possible you will have to do some tweaking to get it to work in your environment. In particular, for example, the commands contain references to specific image resolutions. In addition, note that the commands are run using Linux "sudo" at the World Finals. That is, the actual command used is

```
exec /usr/bin/sudo -u camera -H vlc_command > /dev/null 2>&1 &
```

where *vlc_command* represents the VLC command shown below.

In any case, the commands shown below will hopefully be enough to point you in the proper direction.

Team WebCam Streaming The following single-line command is used at the World Finals to start a stream on port 8080 containing the team's web camera output:

```
vlc -I dummy -q v4l2:///dev/video0:width=1280:height=720:aspect-ratio="16:9"
  :input-slave=alsa://plughw:1,0 --sout '#transcode{venc=x264{keyint=15},vcodec=h264,
  vb=0,scale=0,fps=30,acodec=mpga,ab=128,channels=2}:http{mux=ts,dst=:8080}'
```

Team Desktop Streaming The following VLC command is used at the World Finals to start a stream on port 9090 showing the team's screen:

```
vlc -I dummy -q screen:// --screen-fps=30 --sout
  "#transcode{venc=x264{keyint=15},vcodec=h264,vb=0}:http{mux=ts,dst=:9090/}"
```

Deployment Under Other Application Servers

The CDS distribution comes packaged with an embedded Liberty application server. The CDS application component of this distribution is contained in the file *CDS.war* in the *wlp/usr/servers/cds/apps* folder.

In principle it should be relatively easy to "port" the CDS to a different application server by installing the *CDS.war* file in the new application server according to that server's rules (along with configuring the new application server in ways analogous to the settings described earlier in this document -- for example, specifying the location of the *cdsConfig* file to the new application server).

However, as of this writing we have not explicitly attempted such a port. We would be very interested in hearing about user experiences in porting the CDS to other application servers, especially common servers such as *Tomcat*, *Jetty*, or *JBoss/WildFly*.

If you have any experience in attempting such a port, please feel free to update us about it by sending mail to *icpctools-dev@ecs.csus.edu*.