



ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST



California State University, Sacramento's

PC²
Version 9.2

Contest Administrator's Installation and Configuration Guide

<Last Update: 2011-09-06>

Table of Contents

1	Introduction	5
1.1	Overview.....	5
1.2	Differences From Past Versions.....	6
1.3	References	7
2	PC² Startup Checklist for Geniuses	8
3	Instructions For The Rest Of Us.....	9
3.1	Installation.....	9
3.2	Network / Firewall Requirements.....	9
3.3	Uninstall.....	10
4	PC² Initialization Files.....	11
4.1	pc2v9.ini file	11
5	PC² Startup Procedure	14
5.1	Built-in Commands	14
5.2	Server Startup	14
5.2.1	Non-GUI Server Startup	17
5.3	Server GUI Controls.....	17
5.3.1	Adding Sites	18
5.3.2	Restarting / Reconnecting Servers.....	20
5.3.3	Connections and Logins	21
5.3.4	Replay	21
5.3.5	Additional Server GUI Controls	21
5.4	Starting Clients	22
6	Configuring the Contest in PC²	23
6.1	Administrator Login.....	23
6.2	User Accounts	24
6.2.1	Account Creation.....	24
6.2.2	Account Names and Passwords.....	25
6.2.3	Loading Account Data	27
6.2.4	Importing ICPC Data	28
6.3	Security Alerts	29
6.4	Contest Problems	30

6.4.1	Defining a Problem.....	30
6.4.2	Problem Dialog controls	33
6.4.3	Defining a Problem Validator.....	33
6.4.4	Defining Automated (Computer) Judging	33
6.4.5	Adding a Validator to a Problem.....	34
6.4.6	Changing problem to Computer Judging.....	34
6.4.7	Assigning Auto Judging to Judge module.....	35
6.4.8	Starting Auto Judging	37
6.4.9	Stopping Auto Judging	37
6.5	Contest Languages.....	38
6.5.1	Defining a Language	38
6.5.2	Command Parameter Substitutions.....	40
6.5.3	Language Definition Examples.....	40
6.5.4	Language Definitions In Multi-Site Contests.....	43
6.6	Contest Judgments	45
6.6.1	Defining a Judgments.....	45
6.7	Notifications (Balloons).....	47
6.7.1	Defining a Notification.....	48
6.7.2	Email Server Advanced Settings	49
6.8	Contest Profiles	49
6.9	Options (Settings tab)	50
6.9.1	Edit Scoring Properties.....	51
6.10	Sites	52
7	Starting the Contest.....	53
7.1	Clock Control	53
7.2	Contest Length	54
7.3	Multi-Site Clock Control	55
7.4	Practice Sessions: Resetting A Contest.....	57
8	Monitoring Contest Status	58
8.1	Team Startup Status	58
8.2	The Runs Display.....	59
8.3	Editing Runs.....	60
8.3.1	Extracting Run.....	62
8.4	Filtering Runs.....	63
8.5	Clarifications	65
8.6	Reports	65
9	The PC² Scoreboard	69

9.1	Overview.....	69
9.2	Starting the Scoreboard.....	69
9.3	Scoreboard Updates.....	70
9.4	Scoreboard HTML Files.....	71
9.5	Scoring Groups.....	72
9.6	Scoring Algorithm.....	73
9.7	Adding new HTML Files	74
9.8	Export Data File	75
10	Shutting Down.....	76
10.1	Shutting down a server	76
10.2	Shutting down all servers	76
Appendix A – pc2v9.ini Attributes.....		77
Appendix B – Networking Constraints		79
Appendix C – PC² Server Command Line Arguments		80
Appendix D – ICPC Import/Export Interfaces.....		82
Appendix E – Validators		86
Appendix F – Language Definitions.....		95
Appendix G – Using the PC² API.....		100
Appendix H – Frequently Asked Questions		101
PC² FAQ and Troubleshooting Guide.....		101
10.3	Version 9 Questions	101
10.3.1	Resetting A Contest	101
10.3.2	Connectivity.....	101
10.3.3	Documentation	101
10.4	Version 8 Questions	102
10.4.1	Java.....	102
10.4.2	Contest Configuration.....	102
10.4.3	PC ² Scoreboard.....	102
10.4.4	PC ² Validators (Automated Judging).....	102
10.4.5	Networking	103
10.4.6	Microsoft Windows-Specific.....	103
10.4.7	Other PC ² Questions.....	103
10.4.8	Getting Help	103

10.5 Java.....	106
10.6 Contest Configuration	107
10.7 PC ² Scoreboard.....	109
10.8 PC ² Validators (Automated Judging)	111
10.9 Networking	111
10.10 Microsoft Windows-Specific.....	112
10.11 Other Questions.....	112
10.12 Getting Help	113
Appendix I – PC² Distribution Contents	115
Appendix J – Log files	116
Appendix K – Acronyms, Terms, Definitions	117
Appendix L – Reports Program	118
Appendix L – Troubleshooting / Getting Help	120

1 Introduction

1.1 Overview

PC² is a dynamic, distributed real-time system designed to manage and control Programming Contests. It includes support for multi-site contests, heterogeneous platform operations including mixed Windows and Unix in a single contest, and dynamic real-time updates of contest status and standings to all sites. This manual describes the steps required to install, configure, and run a contest using PC². Further information on PC², including how to obtain a copy of the system, can be found at <http://www.ecs.csus.edu/pc2>.

PC² operates using a client-server architecture. Each site in a contest runs a single PC² *server*, and also runs multiple PC² *clients* which communicate with the site server. Logging into a client using one of several different types of PC² accounts (Administrator, Team, Judge, or Scoreboard) enables that client to perform common contest operations associated with the account type, such as contest configuration and control (Administrator), submitting contestant programs (Team), judging submissions (Judge), and maintaining the current contest standings (Scoreboard).

PC² clients communicate only with the server at their site, regardless of the number of sites in the contest. In a multi-site contest, site servers communicate not only with their own clients but also with other site servers, in order to keep track of global contest state. The following communication requirements must therefore be met in order to run a contest using PC²: (1) a machine running a PC² server must be able to communicate via TCP/IP with every machine running a PC² client at its site; and (2) in a multi-site contest, every machine running a PC² server must be able to communicate via TCP/IP with the machines running PC² servers at every other site. In particular, there must not be any firewalls which prohibit these communication paths; the system will not operate if this communication is blocked¹. It is not necessary for client machines to be able to contact machines at other sites.

Each PC² module (server or client) reads one or more initialization files when it starts; these files are used to configure the module at startup. The client module also tailors its configuration when a user (Team, Judge, etc.) logs in. In a typical PC² contest configuration, each Team, Judge, etc. uses a separate physical machine, and each of these machines runs exactly one client module. It is possible to have multiple clients running on the same physical machine, for example by having different users logging in to different accounts on a shared machine. In this case, each user (Team, Judge, etc.) will be executing their own “Java Virtual Machine (JVM)”, and must have their own separate directory structure – including their own separate copy of the PC² initialization files in their account.

Setting up and running a contest using PC² involves the following steps: (1) installing Java and PC² on the contest machines; (2) creating/editing the necessary initialization files; (3) starting the server(s) and clients(s); (4) configuring PC² for the contest via an Administrator client; and (5) starting the contest so that users (Teams and Judges) can log in. These steps are listed in checklist form in the next chapter, and are described in detail in the remainder of this manual.

¹ See the Appendix titled “Networking Constraints” for further details on using PC² over networks.

1.2 Differences From Past Versions

While the basic functionality of PC² Version 9 (V9) is the same as that of Version 8 (and prior versions), a number of important enhancements have occurred in Version 9. Some of the major enhancements are listed below; these are described in detail in the corresponding section of this manual.

- **Streamlined “.ini” file usage.** The “sitelist.ini” and “reject.ini” files of PC² Version 8 are no longer necessary (the values formerly specified via these files are now configurable interactively). There is now just one single file used to configure the system: “**pc2v9.ini**”.
- **Enhanced Network Configuration.** Each PC² server in a multi-site contest must have a publicly accessible IP address which can accept inbound connections on a single port. Each client must have network access to its server. Network operation behind NAT boxes is also now supported provided the above two requirements are met. It is no longer necessary to open a large range of firewall ports to use the system.
- **Single-point Contest Administration.** In Version 9 it is possible to manage an entire multi-site contest from one site using a single Administrator. All contest details, including generation of PC² accounts for all sites, can be done from a single Admin client; this is described in more detail below.
- **Enhanced Security.** Information transmitted over the network in V9, both between clients and servers and between different servers in a multi-site contest, is now fully encrypted. In addition, contest information stored on disk is also automatically encrypted.
- **Server Control.** When a PC² V9 server is started it displays a “login GUI” similar to that of the clients. The login account name for the first server in a contest is always “**site1**”, with a default password of “**site1**”. Following the very first login the server prompts for a *contest password*. At this point the contest administrator must choose and enter a password which will then be used to perform disk file encryption. If a contest is shut down and restarted it will then be necessary to reenter the same contest password when the server is restarted. (Note that the *contest password* is unrelated to the *site login/password*.) In the case of a multi-site contest, only the *first* server prompts for a contest password; subsequent servers obtain the contest password via their secure connection to the remote server. Starting with PC² Version 9.2, the display of the server GUI can be suppressed using appropriate command-line options.
- **Administrator Options.** The contest administrator has a wide variety of options available in V9, including configuration of user accounts, associating team accounts into groups, specification of judge’s responses to submitted runs, configuration of contest languages and problems, control of the contest clock and other settings, and generation of a variety of contest information reports. Beginning with PC² Version 9.1 the administrator can also configure contest problems for “automated” (computer) judging with or without human review; can import contest configuration and control data from the ICPC registration database; has “filtering” capability for both runs and clarifications; and can control the “look-and-feel” of client GUIs.

- **Application Programming Interface (API).** Beginning with Version 9.1, PC² provides an externally-accessible API. This API supports creation of arbitrary external clients which can connect to a running contest (provided they have the proper authentication credentials – specifically, a valid account and password). Once connected to a contest, the external client can “register” to listen for the occurrence of a variety of contest events. For example, clients can arrange to be notified whenever a new run is submitted, when a judgment is returned, when certain contest configuration data changes, and a host of other events.

1.3 References

While this manual tries to give a complete description of installing and using PC², the following web references may provide additional helpful information.

PC² home page

<http://www.ecs.csus.edu/pc2>

PC² Wiki – up to date articles and information about PC²

http://pc2.ecs.csus.edu/wiki/Main_Page

PC² Bugzilla - enhancement and defect tracking and reporting for PC²

<http://pc2.ecs.csus.edu/bugzilla/>

2 PC² Startup Checklist for Geniuses

For those people who hate to read manuals and would rather take a chance with a shortcut list, here is a *very terse* summary of the steps necessary to install PC² and get it running. Please note that this is provided as a convenience for geniuses (or gluttons for punishment). The remainder of the manual was written to help everyone else. If you have problems after following this list, *please read the rest of the manual* before sending us a request for help.

- ❑ Install Java (version 1.5 or greater) if not already installed; ensure the Java *bin* directory is in the PATH
- ❑ Install PC² by unzipping the PC² distribution to a directory of your choice
- ❑ Add the PC² *bin* directory to the PATH
- ❑ Edit the pc2v9.ini file to point servers and clients to the server IP:port and to specify the appropriate site server name; put the modified .ini file on every server and client machine
- ❑ Start a PC² server using the command "**pc2server**"
 - Login using the name "site1" and password "site1"
 - Enter a contest password (this contest security password will be required to be reentered for all subsequent server startups)
- ❑ Start a PC² Administrator module using the command "**pc2admin**" and login using the name "root" and password "administrator1"
- ❑ Configure at least the following contest items via the Admin:
 - Accounts (generate the necessary accounts)
 - Problems (create one or more contest problems, specifying the problem input data file if there is one)
 - Languages (create one or more contest languages, specifying the language name, compile command line, executable filename, and program execution command line)
- ❑ Press the "Start ALL" button on the Administrator module **Time** tab
- ❑ Start a PC² client on each Team machine (using the command **pc2team**) and each Judge machine (using the command **pc2judge**) and log in using the Admin-created accounts and passwords.
- ❑ Start a PC² client (using the command **pc2board**) on the Scoreboard machine and log in using the "board1" Scoreboard account/password; arrange for the scoreboard-generated HTML files to be accessible to user's browsers.

3 Instructions For The Rest Of Us

In the event that the preceding checklist is a bit too terse, the remainder of this manual discusses the details of using PC² to configure and run a contest. The first step is to install the necessary software, as described in this chapter. The remaining chapters cover initialization files, starting the system, configuring the system for a contest, starting and monitoring the contest, and using the PC² scoreboard. In addition several appendices cover details of certain topics.

3.1 Installation

1. Install the Java Standard Edition (SE) Software Development Kit (SDK) or Java Runtime Environment (JRE), version 1.5 or later on each machine. The remainder of this manual assumes that “\$JAVAHOME” represents the SDK installation directory. For information on obtaining the Java SDK, visit <http://www.javasoft.com>
2. Ensure “\$JAVAHOME/bin” is contained in the PATH environment variable on each machine (i.e., for each user).
3. Go to the PC² home page (see Chapter 1, under References), navigate to the “Download” page, and download the latest PC² “.zip” or “.tar.gz” file to the directory where you wish to install PC² (this can be any directory of your choice).
4. Unzip the downloaded file, being sure to tell the unzip program to “retain directory hierarchy” and “preserve case sensitivity”. This will create a directory named (for example) **pc2-9.2.0**, which for the remainder of this manual we refer to as the \$PC2HOME directory. The \$PC2HOME directory contains **bin**, **lib**, **doc**, **samps** and other directories, plus a default “**pc2v9.ini**” file (see the following chapter) along with several text files giving basic information such as the system version number. The **doc** and **samps** directories contain the system documentation and a variety of sample scripts, files, and other goodies you might want to examine. The file “**doc/index.html**” can be used to browse the documentation. (NOTE: all files and directories which comprise a PC² distribution (.zip or .tar.gz file) will unzip into the \$PC2HOME directory. See the appendices for a complete description of PC² distribution contents.)
5. Add the PC² “bin” directory (that is, the directory \$PC2HOME/bin) to the PATH environment variable on each machine.

3.2 Network / Firewall Requirements

Here are PC² firewall requirements

1. PC² Clients need to be allowed to outbound connections to their server hence servers need to be open to inbound connections from the clients.
2. Servers need to be allowed to outbound connection to all the other servers
3. Servers need to be open to inbound connections from all the other servers

3.3 Uninstall

To uninstall PC² do the following for each PC² Server and client machine:

1. Use the pc2reset script to remove the contents of any PC² directories
2. Remove the pc2v9.ini file and the **archive** directory.
3. Remove the \$PC2HOME directory and its contents
4. Restore the system environment variables (PATH and CLASSPATH)

PC² itself does not make any changes to any machine locations outside those listed above either during installation or execution. In particular, for example, it makes no entries in the Registry on a Windows machine, nor does it copy any files to locations outside the installation directory or the current working directory in any environment.

4 PC² Initialization Files

When a PC² module (server or client) begins running, it reads an “initialization file” named **pc2v9.ini** from the directory in which it was started.² The Contest Administrator must ensure this file is present and edited as necessary on each machine prior to starting a PC² module.

This chapter describes the initialization files and their contents (note: some default versions of the initialization files are provided with the PC² distribution package; these must be edited as necessary). Further descriptions of initialization files and their contents can be found in the Appendices.

4.1 pc2v9.ini file

Every PC² module reads a file named “**pc2v9.ini**” at startup. By default each module looks for its **pc2v9.ini** file *in the current working directory*; thus for example Team machines would typically need to have a **pc2v9.ini** file in their “home” directory. (Command-line arguments can be used to point to a different location for the **pc2v9.ini** file; see the Appendices for details.)

The **pc2v9.ini** file provides key initialization information to the PC² module. The file is formatted in sections. Each section starts with a section-name in square brackets. The following different section names are recognized: **[server]** and **[client]**. Each PC² module reads the entire file, but silently ignores any information which does not pertain to it (for example, server modules ignore data in all sections except the **[server]** section, etc.) All lines starting with “#” or “;” are comments and are also ignored, as are blank lines.

Each section is made up of lines containing "attribute assignment" statements of the form

attributeName=value

The “**attributeName**” is a predefined string chosen from list of PC² configuration attributes. The “**value**” is the value to which the corresponding attribute is set when the **pc2v9.ini** file is read by a module. No spaces are allowed in front of the “value” after the equal-sign.

Some attribute assignment statements are specific to particular sections and have no meaning for other sections (or for the modules that read them). Other attribute assignments are relevant to multiple sections/modules and can appear in different sections.

A complete list of the predefined attributes is given in the Appendices. Most attribute assignments are optional and default values are used if an attribute assignment is not present in the **pc2v9.ini** file. However, certain attributes *must* be specified in the **pc2v9.ini** file in order for the system to function properly.

² Older versions of PC² also read files named **reject.ini** and **sitelist.ini**. However, the functionality provided by those files is incorporated into interactive screens in the Administrator client in PC² V9, and their use is deprecated and scheduled for removal in a future version of the system.

Based on these rules, a minimum sample `pc2v9.ini` file is shown below. Note that since all modules ignore sections and attributes which do not apply to them, it is permissible to create a *single* `pc2v9.ini` file containing all the required entries and put this same file on all machines at a given site.

```
# sample pc2v9.ini file

[client]
# Tell PC2 clients where to find their server (IP and port)
server=198.51.100.50:50002
```

The sample `pc2v9.ini` file shown above would be appropriate for both client and server machines in a single-site contest.

In a multi-site contest, the server at one of the sites is designated the “primary server” and servers at all other sites are designated “secondary servers”. When a primary server is started, it waits for other servers to contact it. When a secondary server is started, it automatically attempts to contact the primary server; this is how the inter-server communication in a multi-site contest is established. (The distinction between whether a server waits to be contacted (is a primary) or initiates remote contact (is a secondary) is in fact the only distinction between “primary” and “secondary” servers.)

In a multi-site contest exactly one of the servers should be started as a primary server (it does not matter which site has the primary server; once communication is established all sites run as peers). The servers at all other sites should be started as secondary servers. Designation of a server as primary or secondary is controlled by the contents of the `pc2v9.ini` file.

By default (that is, in the absence of any information in the `pc2v9.ini` to the contrary), a server assumes it is a primary server when it starts. Designating a server as a secondary server is accomplished by providing an additional entry in the `[server]` section of the `pc2v9.ini` file of that server. This additional entry, known as the *remoteServer attribute*, tells the secondary server the IP address and port number at which it should attempt to contact the primary server. If this `remoteServer` attribute is *not* present in `[server]` section of the `pc2v9.ini` file when a server starts, the server implicitly assumes it is a primary server.

Thus for example, the `pc2v9.ini` file on the primary server in a multi-site contest might look like the sample above (since it does not contain any `remoteServer` attribute), whereas the `pc2v9.ini` file on machines at a second site might look like:

```
# sample pc2v9.ini for a second site

[client]
# tell clients where to find their site's server (IP and port)
server=203.0.113.7:50002

[server]
# Tell this (secondary) server how to contact the primary server
remoteServer=198.51.100.50:50002
```

Note that the (sample) IP address given in the **[client]** section of the above **pc2v9.ini** file for a secondary site is the (hypothetical) IP address of the server for that site, whereas the IP address given in the **remoteServer** attribute in the **[server]** section is the address of the primary server – the address which the (secondary) server should use to contact the primary server and “join the contest”.

5 PC² Startup Procedure

5.1 Built-in Commands

Once PC² has been installed and the necessary “.ini” files have been properly set up (i.e., edited and placed in the appropriate startup directory), the normal PC² startup procedure is to start a primary server, then start an Admin client connected to that server and use the Admin client to configure the contest details (problems, languages, etc.). Once the contest has been fully configured using the Admin client, secondary servers can be started at remote sites (if any), followed by additional clients at both the primary and (in the case of a multi-site contest) secondary sites.

The PC² distribution comes with a collection of “command scripts” designed to simplify starting the various modules. The available command scripts and their corresponding functions are listed below.³ To invoke the specified function, simply type the corresponding command at a command prompt. The commands reside in the “bin” directory beneath “\$PC2HOME” (the root directory of an unzipped PC² installation), so the normal method of invoking them would be to change to the contest directory (that is, the directory where the contest will be run from, where logs are to be kept, etc.), and type the command name (note that this assumes, as previously recommended, that the \$PC2HOME/bin directory has been added to the \$PATH).

Command	Function
<code>pc2server</code>	Starts a PC ² Server
<code>pc2admin</code>	Starts a PC ² Client expecting an Administrator login
<code>pc2team</code>	Starts a PC ² Client expecting a Team login
<code>pc2judge</code>	Starts a PC ² Client expecting a Judge login
<code>pc2board</code>	Starts a PC ² Client expecting a Scoreboard login

Thus the normal startup procedure would be to invoke the command “`pc2server`” to start a server, then *in a separate command window* to invoke the command “`pc2admin`” to start an Administrative client to be used to configure the contest details in PC². Once the contest details are configured, other clients can be started (as well as servers at other sites) using the appropriate commands.

5.2 Server Startup

When a server is started (using the command “`pc2server`”), the user will see a login window similar to the following:

³ In a Windows environment the command scripts are all “batch files” whose names correspond to the given command name followed by “.bat”. In a Unix environment the command scripts are all Bourne Shell scripts whose names match the given command names. Thus the same command name can be used regardless of the underlying OS.

PC^2 Login

California State University, Sacramento
Programming Contest Control System

SACRAMENTO STATE

Name

Password

Login Exit

PC^2 version 9.2 20110720 2302

acm International Collegiate Programming Contest IBM event sponsor

The default login name for the primary (first) site server is **site1**; the default password is also **site1** (the password can subsequently be changed; see below). Entering these values and pressing the Login button on the above screen will initiate the login process. However, there is a second step which must be performed to complete the login process.

At the time a server is started for Site 1 for the *very first time*, there is no contest-specific information stored in the system. All contest information which is subsequently entered will be stored in *encrypted* form, to protect the integrity of the contest data. In order to manage the encryption and allow authorized access at a later time, the Contest Administrator must provide a *contest master password*. Thus, on the very first login to a primary (first) server, the following screen will appear:

Set Contest Master Password

The Contest Password

In order to insure contest security, all contest data is protected by a master password. Before anyone can restart a contest or access sensitive data they will be asked to enter this contest master password. This screen is the place where you set (and confirm) the value of the contest master password. (Note that the contest master password is independent of the passwords needed to login to any specific contest account -- Server, Admin, Team, Judge, etc.)

Contest Password

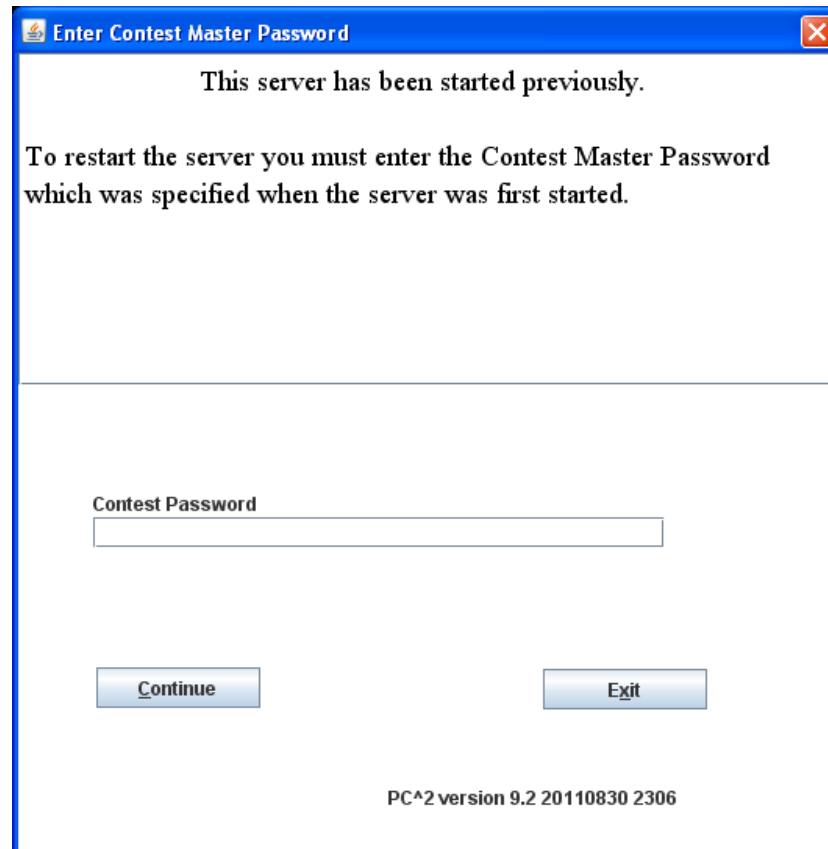
Confirm Contest Password

Continue Exit

PC^2 version 9.2 20110720 2302

The Contest Administrator must enter (and confirm) a password to be used to perform contest data encryption. Note that *this password provides access to all contest-related data*; it should be well-chosen and well-protected. Note also that *there is no default value for this password*; it must be set (chosen and confirmed) by the Contest Administrator. Entering and confirming a contest master password (in addition to the login ID and login password) completes the login process.

If a first-site server is started again at some later point in time (i.e. after a contest master password has been set), pressing the Login button on the server's main login screen displays the following screen:



At this point the user must enter the Contest Password in order to complete the login process.

Only the primary (first) server in a contest prompts for the contest master password. In a multi-site contest the user starts a (secondary) server using the **pc2server** command as above, and logs in using the default site name (for example, **site2**) and password (**site2**). The secondary server contacts the primary server (pointed to by the **remoteServer=xxx** entry in its **pc2v9.ini** file) and obtains the current contest master password directly from the remote server. Note that the second server does *not display* the obtained contest master password; if people at secondary sites need this password for some reason then they must obtain it from the Contest Administrator at the primary (first) site.

5.2.1 Non-GUI Server Startup

In some environments it may be desirable to run a PC² server without a GUI front-end. This can be accomplished using the `--nogui` command line argument described in the appendix on Server Command Line arguments. Note that if this option is chosen there is no GUI-based way to enter login, account password and contest password information. These may be entered on the command line as well. For example, to start a primary (first) server with no GUI the following command could be used:⁴

```
pc2server --nogui --login site1 --password site1 --contestpassword contest
```

To start a secondary server without a GUI in a multi-site contest the following command could be used:⁴

```
pc2server --nogui --login site2 --password site2
```

Note that since this server is presumed to be secondary (meaning it has a `pc2v9.ini` entry pointing to a remote server which is the primary server), the `--contestpassword` option is omitted. This is because secondary servers obtain the contest master password directly from the primary server, as described above.

When a server is started using the `--nogui` option, it sends all of its output in text form to the console window from which it was started. The following shows an example of a server started in this way. Any subsequent text produced by the server (e.g. error messages or informational text) would appear on subsequent lines on the console.

```
CSUS Programming Contest System
Version 9.2 20101008 (Friday, October 8th 2010 20:25 UTC)
Java ver 1.6.0_11 build 2182 Windows Vista 6.1 (x86)
Build 2182
Date: 10/8/10 3:51 PM
Working directory is C:\pc2-9.2-2182

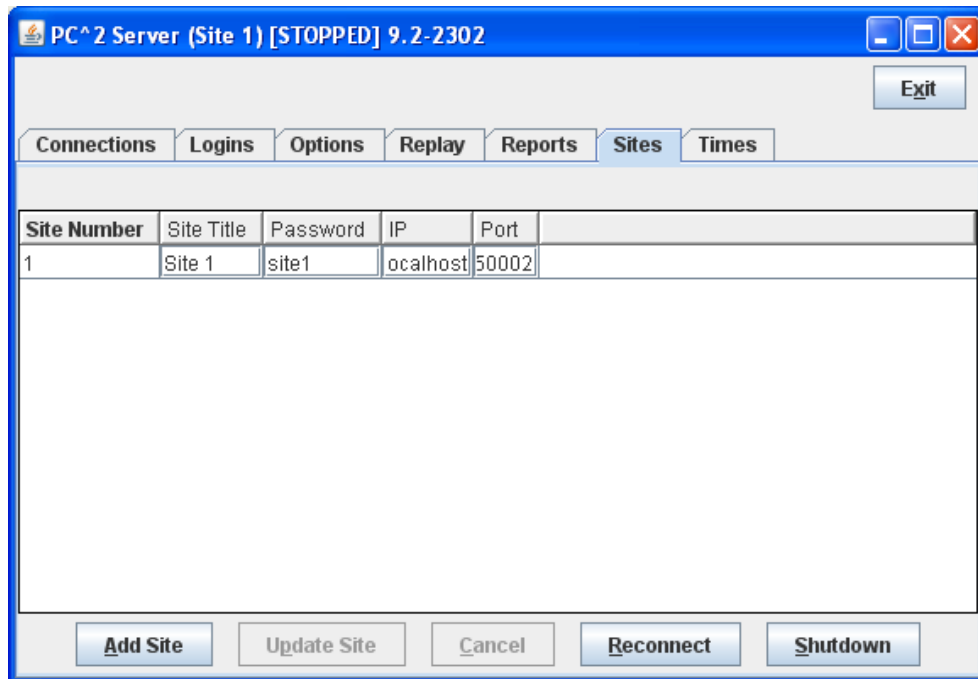
Fri Oct 08 15:51:59 PDT 2010 server (Site 1 - Site 1) started
```

To halt a non-GUI server, use the Shutdown button (on the Site tab on the Administrator) to gracefully halt the server.

5.3 Server GUI Controls

Upon successful login to a server (assuming the server was started without the `--nogui` option) a GUI similar to the following appears:

⁴ Note that a drawback of this approach is that the login and contest master passwords must be typed in plain-text on the command line. A much more secure option is to take advantage of the “-F” command line argument, which allows specification on the command line of a *file* containing the necessary security information. See the appendix titled “PC² Server Command Line Arguments” for details.



The tabs across the top of this GUI allow the user to examine and modify various configuration items in the contest. Those items which are primarily server-related are described below; some of the tabs also appear in the PC² Administrator GUI and are described in the chapter on “Configuring the Contest”.⁵

5.3.1 Adding Sites

The **Sites** pane (shown above) lists each contest site which the system knows about. Initially only “Site 1” is known; in order for a server at another site to join the contest, the additional site must first be added to the Sites list.

To make the system aware of the presence of another site, press the **Add Site** button to create a new row in the grid.⁶ In the new row which appears, select the **Password** field and *change the password from the default value.*⁷ Next, select the **IP** field in the new row and enter the IP address for the new site (that is, the IP address of that site’s server machine); then select the **Port** field in the row and enter the port number at which the new site is expected to be contacting

⁵ In the case of starting a server with the “`--nogui`” option the PC² Administrator client is the *only* way to access some of these screens. In addition, some server-specific capabilities can *only* be accessed via the server GUI.

⁶ The Add Site button also appears on the **Sites** tab in the Administration module GUI (where it can be found under the **Run Contest** tab).

⁷ It is *critically important* for the security of your contest that you enter *new passwords* for *every* site (including the first one). Otherwise, since the default values for site passwords are well-known (published for example in this manual), some bad-guy could *start his own server and connect to your contest while it is running*.

the primary site.⁸ Optionally, select the **Site Title** field and assign a name to the site. Finally, press the **Update Site** button to save the site info.

Note that adding a site is a two step process: use Add Site to input the site data then Update Site to save the site data.

When starting a server for a site, the user must supply two data values: a login name and a password. For each site in the contest, the server's login name is the word "*site*" followed immediately (no spaces) by the *site number* (for example, *site1* or *site3*). Each server's *site number* is the value given in the leftmost field in the **Sites** display pane. The *password* for logging in to each site server is the value given in the **Password** field. If the password for a remote site is changed using the **Sites** pane, the new password must be relayed to the remote site in order for them to be able to log in to the contest.

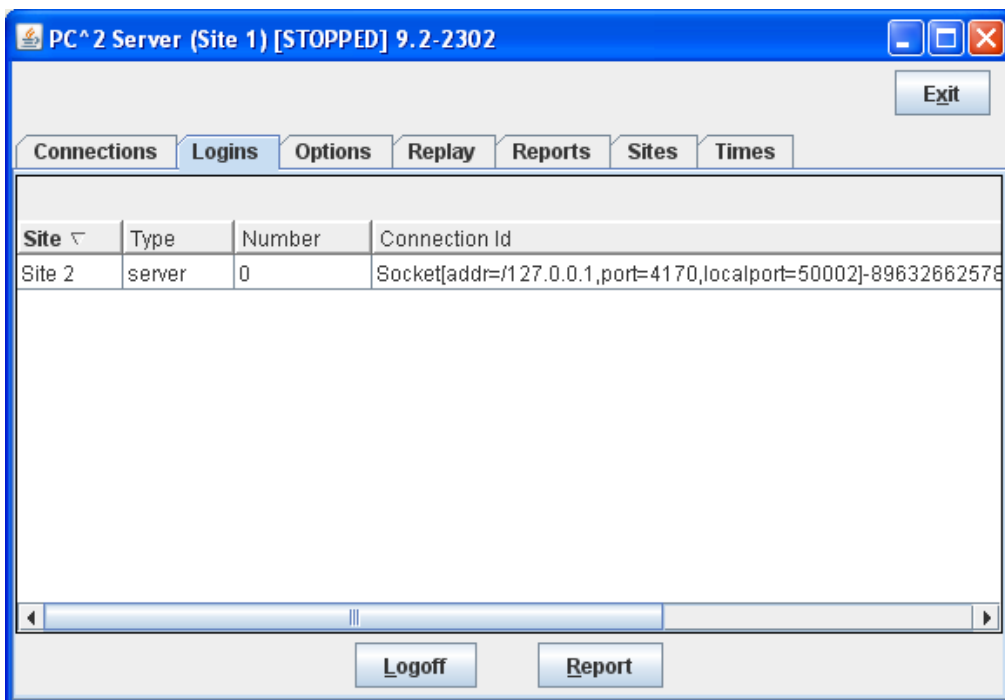
Note that there is no correlation between the value shown in the **Site Title** field of the **Sites** pane and the data required to log a server into a contest; login names for servers are always "*siteX*", where '*X*' is the *site number* shown at the left of each row in the **Sites** pane. The only function of the "Site Title" field is to provide a convenient human-readable reference for each site; that reference string is not used in any internal operations in PC².

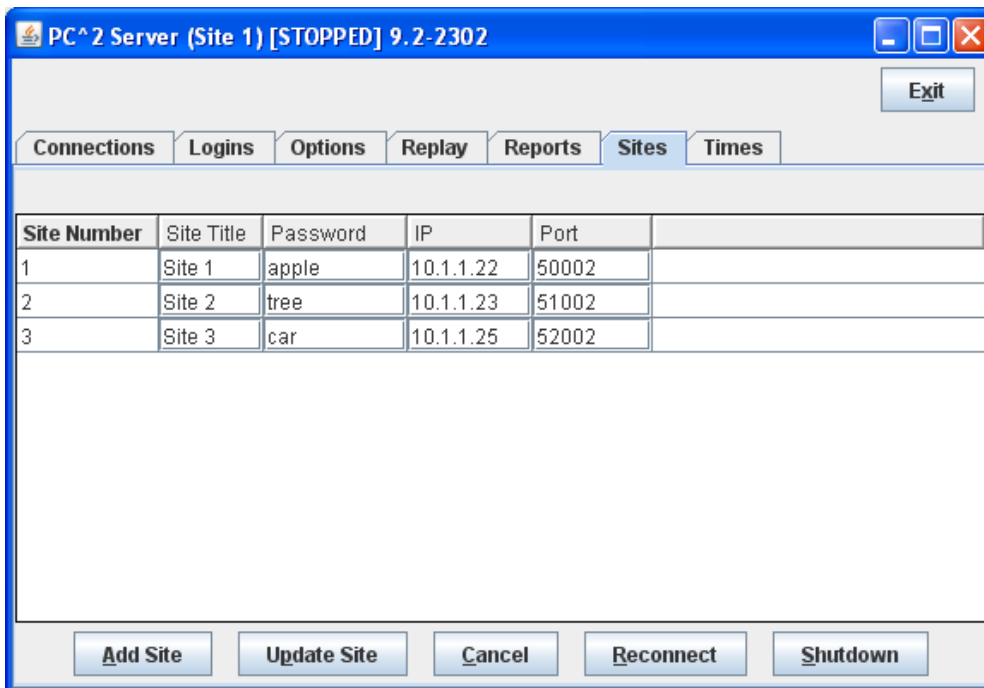
⁸ Care should be taken to set the port number to the port which will be specified in the **pc2v9.ini** file at the remote site; the servers will not be able to communicate with each other if the port numbers do not match. Also note that when a new "site row" is added to the grid using the **Add Site** button, the default port number assigned to the new site is *not* the same as the default for the first site. Typically the port numbers must be changed to match (assuming all sites will use the same port for communication).

5.3.2 Restarting / Reconnecting Servers

During a contest there may be a loss of connectivity between sites, or a situation where one or more servers get “out of sync” with the others. In PC² Version 8 one had to shutdown (kill) the server and restart it when this happened. In Version 9 there is a *Reconnection* feature available to help with this situation.

To determine whether a site to which communication has been lost can be reconnected without killing and restarting the site’s server, check the **Logins** pane (see below). If the server from that site appears in a row in the **Logins** grid, there is a connection between the sites. In this example on Site 1, Site 2 is logged in.





To reconnect a site that was previously logged in⁹ but is now disconnected, use the Reconnect button. Note that the Reconnect button is on the Sites tab not the Login tab. When the site is reconnected, that site will appear in the list on the Login tab.

5.3.3 Connections and Logins

These two tabs on the server GUI show what network connections and what PC² client logins have occurred since the server was started. They can sometimes be used to help with reconnection when sites get disconnected, or to trace anomalous or error conditions in the system. They can also be used to force a disconnection or logout from the system; this is done by highlighting (selecting) row in the corresponding grid and pushing the **Disconnect** (or **Logoff**) button.

5.3.4 Replay

This tab contains code designed to allow reading a contest events database and recreating the sequence of contest events within PC². Currently this code is experimental and should be used with extreme caution. Contact the PC² Team for more details about this feature.

5.3.5 Additional Server GUI Controls

The remaining tabs on the Server GUI (**Options**, **Profiles**, **Reports**, and **Time**) are replicated on the Administrator GUI and are described in the chapter on Configuring the Contest.

⁹ To reconnect a site the site must have previously been started. Reconnection will not restart a remote server.

5.4 Starting Clients

Once a PC² server is running at a site, users (Contest Administrators, Judges, and Teams) at the site can start PC² clients to login and use the system¹⁰. The normal procedure is first to start a client using the “`pc2admin`” command and login as the “root” administrator (password “administrator1”) in order to configure the contest. Subsequently each Contest Judge would start a client using the “`pc2judge`” command, and each Team would start a client using the “`pc2team`” command. The Contest Administrator would normally also start a PC² scoreboard using the “`pc2board`” command, logging in using the PC² account “board1”.

Each time a client is started, the client will read its `pc2v9.ini` file to determine its site name and the location of its server, and then contact the server. Following this initialization sequence, the client will display a “login” window as shown below, indicating that it is ready to accept a user (Team, Judge, Administrator, or Scoreboard) login. Depending on the logging levels specified in the client’s `pc2v9.ini` file, the progress of these steps will be displayed and/or written to the file `pc2.log` in the client’s startup directory. If any errors occur or the client fails to produce the login screen, check the log file in the `logs` directory for more details.¹¹



¹⁰ In the case of users logging in to a server (e.g. via Xterminals under Unix) rather than where each user has their own machine, each user must start a client on the server via their terminal window. Each client must be started in its own separate directory, which must contain the appropriate initialization files. Under Xwindows, the `DISPLAY` environment variable can be used to direct PC² graphical output from the client back to the Xterminal.

¹¹ See the Appendices for further descriptions of log files and their contents.

6 Configuring the Contest in PC²

6.1 Administrator Login

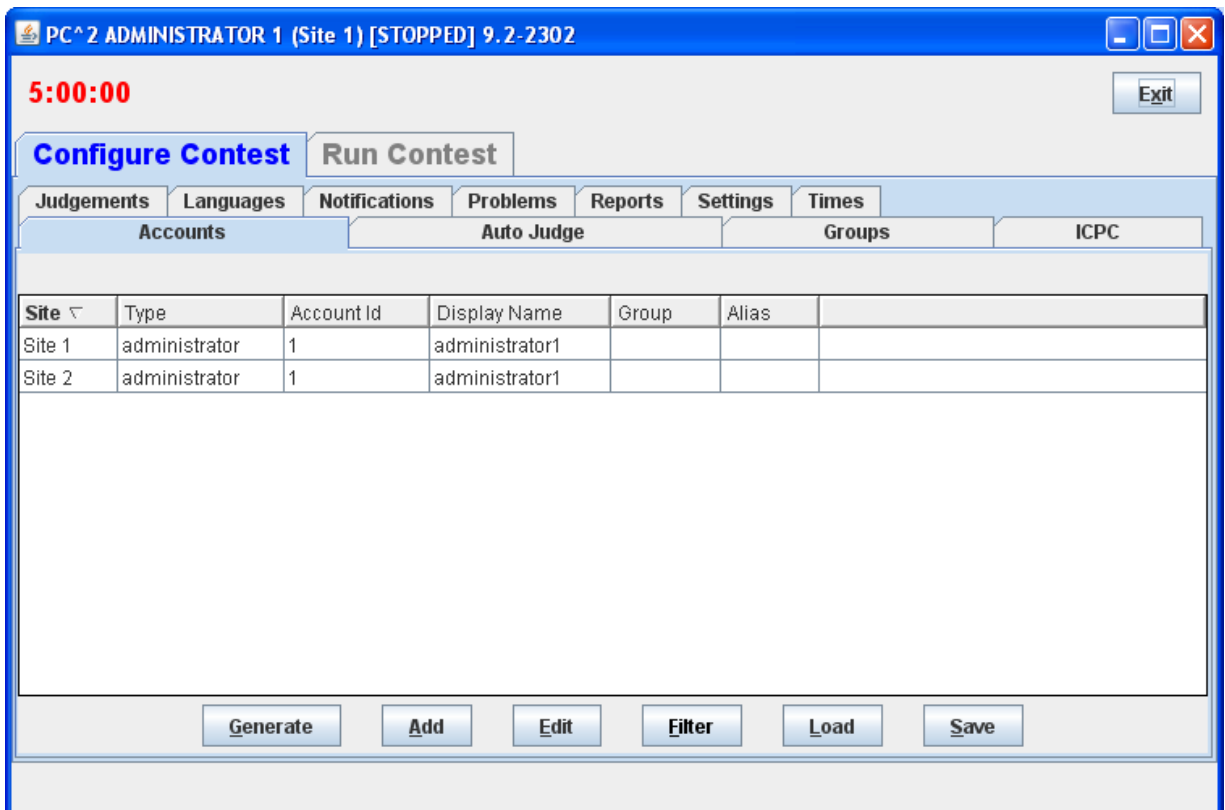
Once PC² is set up and running, it is necessary to “log in” to the system via a client login window in order to use the system. A PC² “account” is required in order to log in. Initially only a *single* account exists; the account name (“login ID”) of this single account is “**root**”. This account is a *master Administrative account* which is used to configure the PC² system initially for the contest. Regular users (especially Teams) should NOT be given access to this account.

The default password for the **root** account is **administrator1**. Note that the default master password is given right here in this paragraph of this document, which is publicly available on the Web.

Caveat Administrator : change the root password!

Passwords can be changed via the “Manage Accounts” function on the “Accounts” tab; see below.

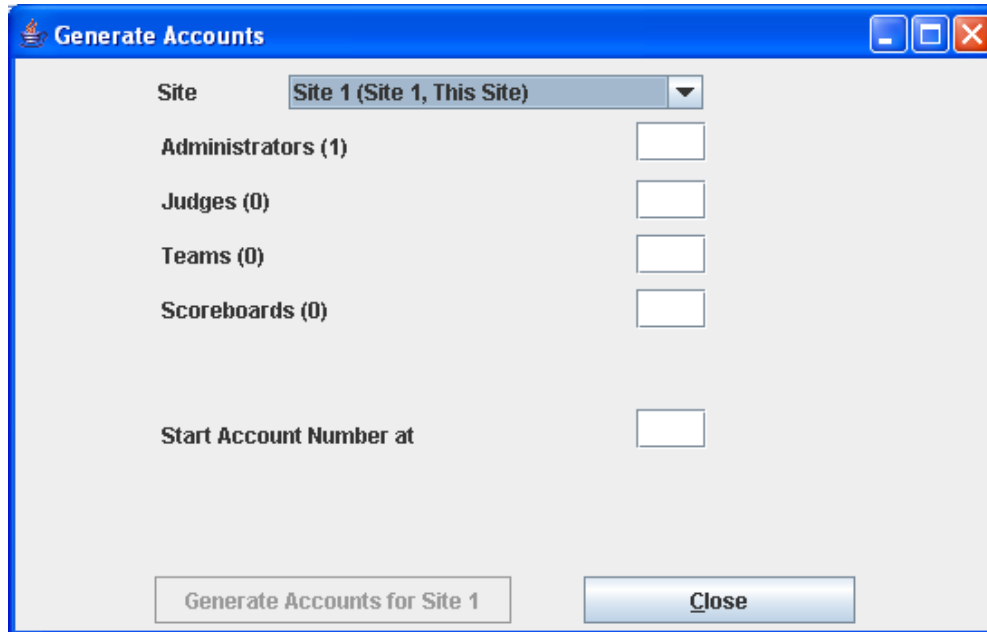
After logging in to the “**root**” Administrator account, the following screen will be displayed. This is referred to as the “main Administrator screen”. It provides a series of “tabs” across the top to select various contest administration functions. The tabs which are used to configure the contest prior to starting are described in the remainder of this chapter. Tabs used to start the contest and monitor its progress are covered in the following chapters.



6.2 User Accounts

6.2.1 Account Creation

Before any logins other than **root** can occur, it is necessary to create user accounts. To create accounts for users, click the **Generate** button on the **Accounts** tab on the Configure Tab on the Administrator screen. This will display the following screen:



Note that the **(1)** to the right of the “Administrators” label means that currently one Administrator account exists – that one account is the **root** account – and no other accounts exist. It is necessary to create one Team account for each team at this site, one Judge account for each person who will be judging the contest at this site, and at least one Scoreboard (“board”) account if the PC² scoreboard is going to be used at this site for tracking contest results. (It does not hurt to generate a few extra accounts in each category, for flexibility.)

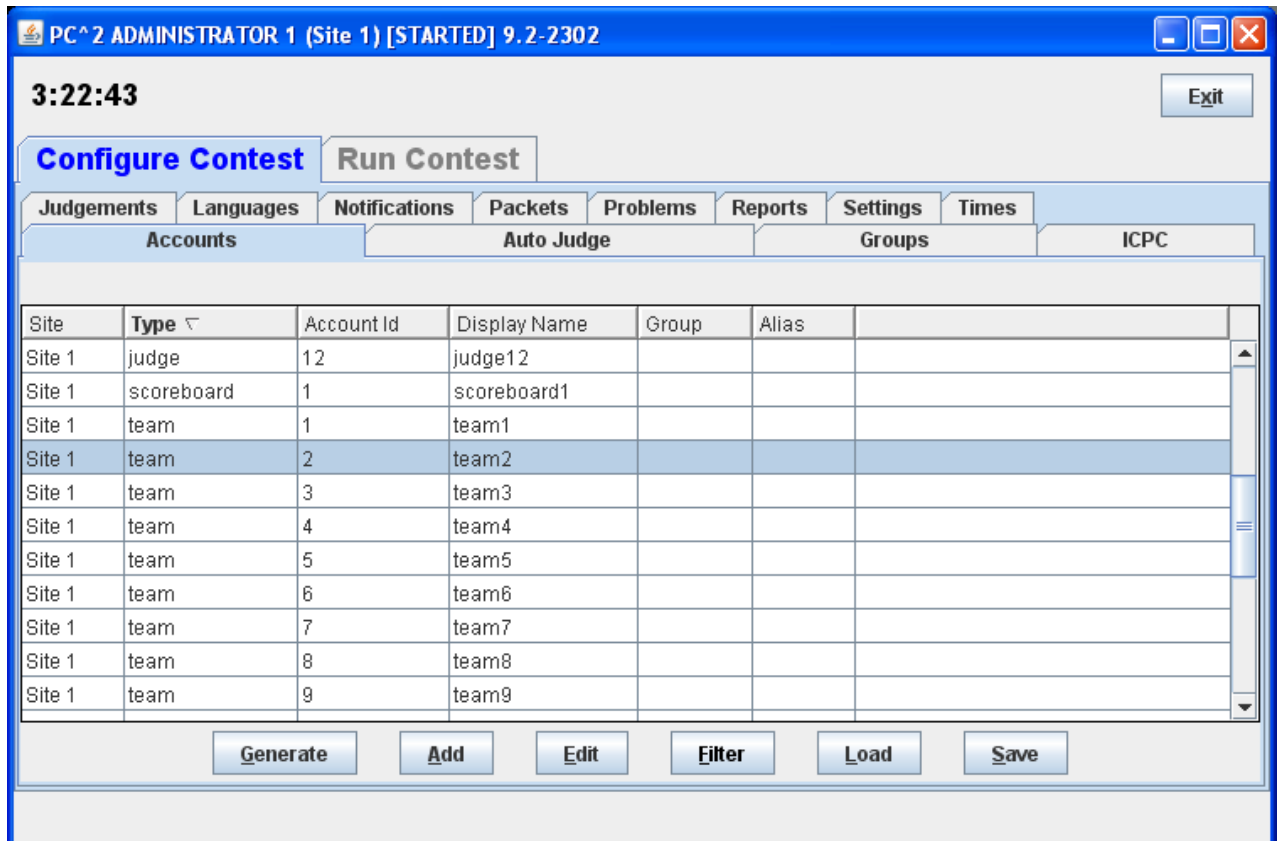
Enter the desired number in each box, then click the **Generate** button. Depending on the number of accounts, number of sites, and communication delays, it will take anywhere from a few seconds to several minutes for the account generation to complete. Once the accounts are generated the system will automatically return to the main Administrator screen. PC² accounts always start with the word **team**, **judge**, **admin**, or **board**, followed by a number. See the next section for information on viewing/changing the state of generated accounts.

Accounts in PC² are *site-specific*. This means that in a multi-site contest an Administrator must create the accounts for each site. There are two steps for the Administrator to create accounts on a site. First, login a server from each of those sites. Second, an Administrator creates those accounts by selecting the site then creating the accounts as described above.

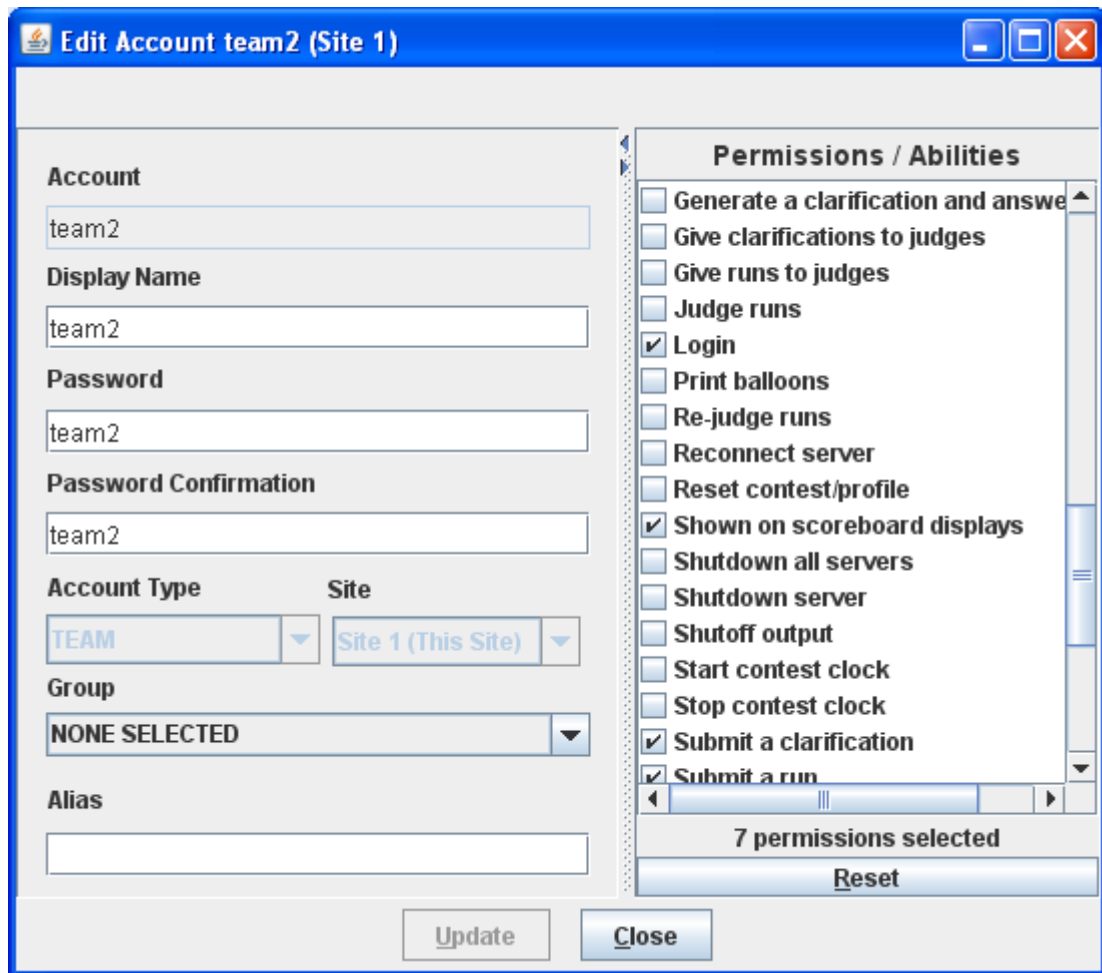
6.2.2 Account Names and Passwords

Each generated account will be created with a password, but at present the default (only available) specification for passwords on newly-generated accounts is “Passwords same as Account Name”. This means for example that the password for the account “team1” is “team1”; the password for the account “judge1” is “judge1”; etc. Each account name and password are created with all letters in lowercase.

Passwords for accounts can be changed from their default values by editing each account. To do this use the Edit button on the Accounts Tab under the Configure Contest tab.



To edit an account, click on the account in the display grid to select it (“team2” has been selected in the display shown above), and then click the “Edit” button. This will display a new “Edit Account” window, shown below:



The “Display Name” for an account is the name which will appear on the PC² Scoreboard; this can be set to any desired value (such as the name of the team’s school, or the team member’s names). The “Password” and “Verify Password” fields can be used to set any desired password for the account.

The Permission “Shown on scoreboard displays” checkbox determines whether a team account will be considered in computing the scoreboard standings; if there are some team accounts which will not be used then you should uncheck their “Shown on scoreboard displays” checkboxes – otherwise they will appear on the scoreboard as teams which have solved no problems. *Note that the “Shown on scoreboard displays” checkbox only determines whether a team appears on the scoreboard; teams which are not “Active” can still log in, submit runs, and otherwise participate in the contest. This is designed to allow “guest” or other “non-competitive” teams to participate. (To prohibit any activity from a team account, change the account password or uncheck the “Login” Permission checkbox)*

The “Group” field is used to associate accounts with different “regions” or “groups”. This is used in conjunction with the PC² scoreboard for displaying rankings of different subgroups (see the chapter on the PC² Scoreboard for further details).

Note that PC² accounts are unrelated to any user accounts which may otherwise exist on the systems being used for the contest (for example, user accounts provided by the operating system).

In a multi-site contest, newly created PC² accounts are automatically distributed throughout the entire system, including across multiple remote sites. As previously noted, accounts are “site-specific”. Note also, however, that accounts at different sites are numbered using the same sequence; the first team account at Site 1 is called “team1”, and the first team account at Site 2 is *also* called “team1”, etc.. Accounts are therefore identified by always giving both the Site number and the Team number, as in “Site1Team1”, which is a *different account* from “Site2Team1”.

6.2.3 Loading Account Data

Since editing account data (e.g. Display Names, Passwords, etc.) interactively for every account is cumbersome, it is desirable to be able to prepare the data “offline” ahead of time and then load it into PC². This can be done by preparing an “account data” file and using the **Load** button on the Accounts tab load the data into the system.

An “account data load file” consists of a series of text lines, a single line that defines the account data fields that will be loaded followed by lines which contain information for each account.

The format of the account data load file is as follows. File lines starting with ! or # in the first column are ignored. Each non-comment line has fields, separated by a <tab> (ASCII 8).

The first line of the file must contain the field names to be loaded. The site and account fields are required, the field names are:

site – site number

- account – team login name (ex. team1, judge4, scoreboard2)
- password – account password
- group – group name
- displayname – display name for account on server
- alias – a alias display name shown to judges
- permdisplay- true or false, display on scoreboard
- permlogin – true or false, allowed to login

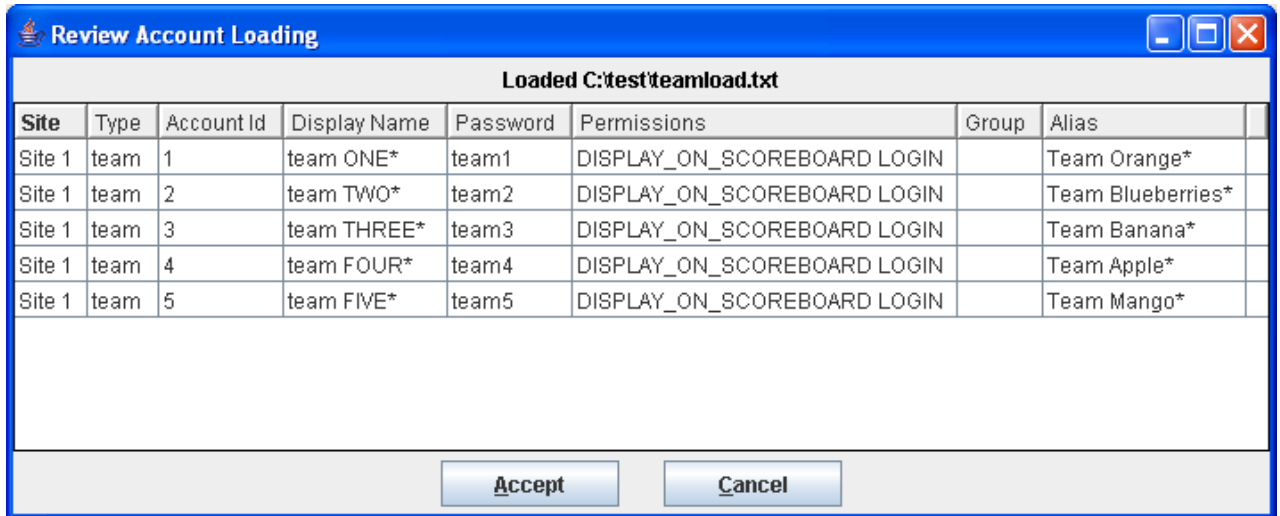
For example, to initialize accounts “team1”, “team2”, and “team3” so that the “team1” account displays on the scoreboard with the name “Number 1” and has a password of “pass1”, while the “team2” account displays on the scoreboard with the name “Team Number 2” and has a password of “myPass”, and the “team3” account is made inactive (does not display on the scoreboard), the following entries would be placed in the load accounts data file for teams:

```
site<tab>account<tab>displayname<tab>password<tab>permdisplay
1<tab>team1<tab>Number 1<tab>pass1<tab>true
1<tab>team2<tab>Team Number 2<tab>myPass<tab>true
1<tab>team3<tab>My School Name<tab><tab>false
```

In the example above to view the tab character the <tab> represents a single tab character (ASCII 9).

Imported values overwrite any values that were in the system previously. Also, it is not necessary to provide a record in the data file for every account; the site and account fields determines which accounts will be modified (all “missing” accounts will remain unchanged).

To load the account load file use the Load button on the Accounts Tab (under the Configure tab). The load button will display a File Open dialog, select the name of the account load file and click Ok, next the Review Account Loading dialog will appear.



Any changes/differences that will be applied will have an asterisk at the end. Click on Accept to apply the changes.

6.2.4 Importing ICPC Data

PC² was designed for supporting the ACM International Collegiate Programming Contest, including its local and Regional contests worldwide. The ICPC maintains an online Contest Registration system which is used by Regional Contest Directors (RCDs) around the world to manage participation in the various ICPC Regional Contests.¹² PC² provides interfaces to import contest registration data from the ICPC Registration system, and also to export contest results back to the ICPC web site. See the Appendix on ICPC Import/Export Interfaces for further information on importing/exporting ICPC Registration system contest data.

¹² Visit the ICPC web site at <http://icpc.baylor.edu/icpc/> for further details.

6.3 Security Alerts

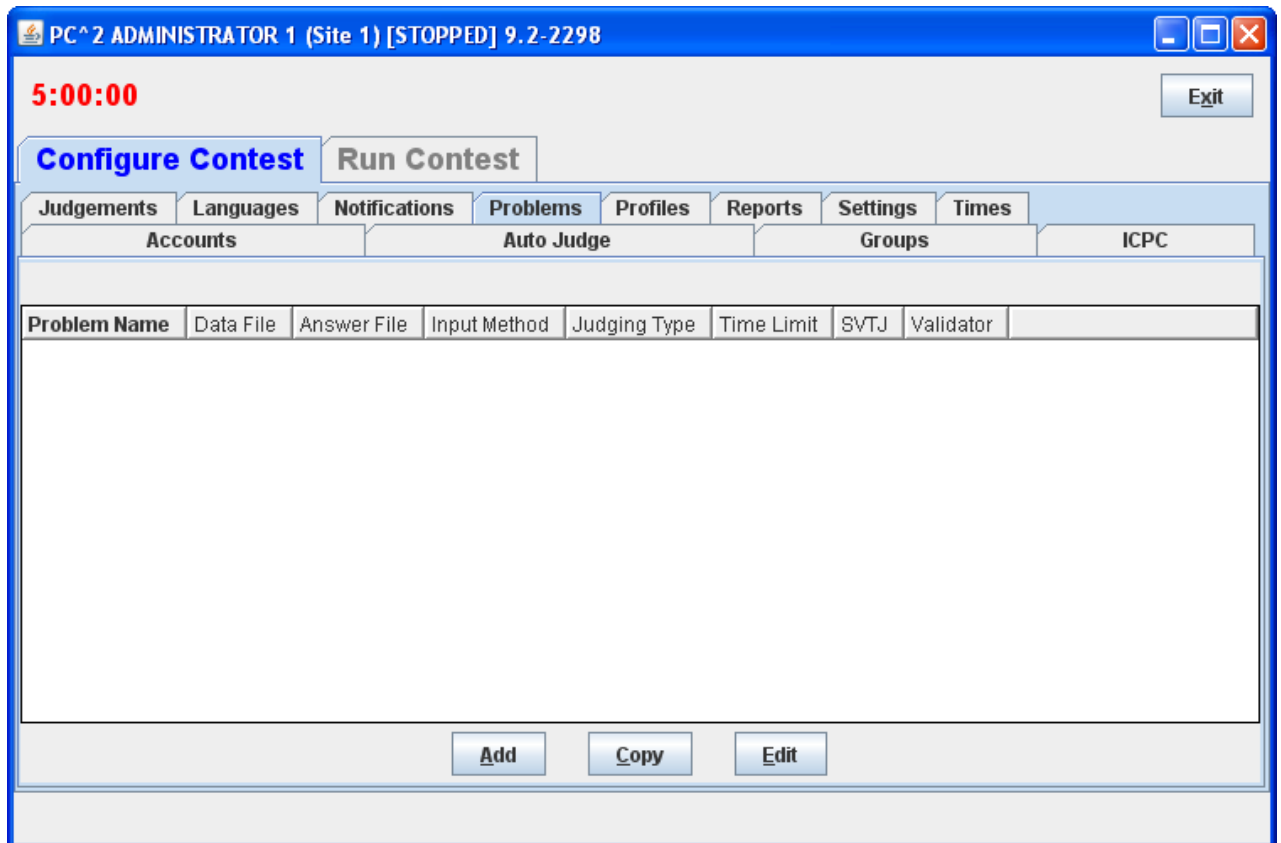
There are a number of security features in PC², disk files encryption, network traffic encryption and security alerts. Version 9 security alert windows will automatically appear on server UI and Admin UI when an PC² client logs in successfully while the same client is already logged in. The security information will also be logged in a security log, in this example the security log file is named: [logs/SERVER0@site1.security-0.log](#) . An example of the log entries is:

```
110729 152323.093|INFO|main|initializeSecurityAlertWindow|Security Log Started
      Version 9.2 20110720 (Wednesday, July 20th 2011 09:30 PDT) Java ver
      1.6.0_20 build 0 Windows XP 5.1 (x86)
110729 152340.312|SEVERE|Thread-10|newMessage|SecurityException From:
      ADMINISTRATOR1 @ site 1 ADMINISTRATOR1 @ site 1: duplicate login request;
      previous login forced off
|edu.csus.ecs.pc2.core.exception.ContestSecurityException: ADMINISTRATOR1 @
      site 1: duplicate login request; previous login forced off
```

6.4 Contest Problems

6.4.1 Defining a Problem

PC² must be provided with information about the problem set to be used in the contest. To enter this information, click on the **Problems** tab at the top of the main Administrator screen. This will produce a display similar to the following:



Note that initially no problems are listed since none have been added to the system. To add a problem, click the **Add** button. This will produce the “**Add New Problem**” dialog shown below.

To define a contest problem to the system, perform the following steps using the **Add New Problem** dialog:

- Enter the problem name in the top textbox.
- If the problem requires an input data set, click the “Problem Requires Input Data” checkbox and then
 1. select either “Stdin” or “File”, depending on whether the problem description tells teams to write their programs to obtain input data from “standard input” or from a file¹³, then

¹³ Note that teams can be instructed to write programs which read from “stdin” even though the administrator provides the input data in a file; PC² arranges that the content of the specified file is available in the current directory at runtime (for the case of reading from a file), or that the content of the file is presented to the program’s standard input channel (if that input selection is specified).

2. use the **Browse** button to select the data file.¹⁴

- If the Judges have provided an “Answer File” (a file showing the expected output of a program correctly solving this problem), click the “Judges Have Provided an Answer File” button and then use the **Browse** button to select the Answer File.
- Click the **Update** button to store the problem information.

As each set of contest problem information is entered, it will be displayed on the main Administrator screen (when the **Problems** tab has been selected). To change some previously entered information for a problem, click on the problem row in the main display to select it, then click the **Edit** button. This will return to the **Edit Problem** dialog, where changes can be made.

The following additional notes apply when entering data using the **Edit Problem** dialog:

- The Run Timeout Limit value (shown as 120 in the sample screen above) is enforced by PC². A count-up timer is displayed during program execution so that the Judge can tell how long the program has been executing, when the specified timeout limit is reached the program will be terminated and the Validator judgment show “No - Time Limit Exceeded” The timer also includes a button to allow the Judge to terminate the program at any time.
- The content of the input data file for a problem is stored internally when the **Update** button is pressed (that is, PC² makes an internal copy of the file). For this reason, editing the user’s copy of the file will *not* automatically change the data presented to team programs. To modify the data file for a problem, the contest administrator must **EDIT THE PROBLEM** and press the **Update** button. Upon pressing the Update button, a prompt will appear confirming that the file has changed on disk. Answer Yes to the prompt to re-load the data file.
- The **Validator** tab on the **Edit Problem** dialog is used to interface an automated judging program for this problem to PC². See the Appendix on Validators for further details.
- All team program output is expected to go to “standard output” (where it is captured by PC² and saved for display to the Judges). More specifically, there is no mechanism in the current version of PC² for dealing with programs which are written to send their output to a destination other than “stdout” (for example, programs which send their output to a file).¹⁵

¹⁴ In the current version of PC², only one input data set is allowed per problem. To test programs against multiple data sets, place all the data sets in a single file, put a “counter” record at the front specifying the number of data sets, and include instructions in the problem description telling teams to process the input data this way. And yes, it’s another thing on our list of desired improvements for a future version of the system...

¹⁵ There is in principle no reason a contest administrator could not use the PC² “Validator” capability to effectively examine and process output sent to a file by a team’s program – including displaying that output for the Judges. While this is not the primary intent of the Validator capability, it could be used as an effective workaround for this limitation. See the Appendix on Validators for details.

- Contest problems in PC² are *global*, in the sense that once a problem definition is entered by the Contest Administrator that problem definition is broadcast to all sites. There is no mechanism for having teams at different sites in a multi-site contest see different descriptions of the same problem. If there is some reason that different descriptions are needed for the same problem (for example, if a problem needs to be described differently at different sites due to OS differences), it is necessary to enter the different problem descriptions effectively as different problems. (While this is not very elegant, it is also something that we *rarely* – virtually never – see in real contests... Said another way, PC² views a contest as a set of teams all working on an identical problem set.) Note also that contest problems must appear in the same *order* at each site in a multi-site contest; see the chapter on Loosely-Coupled Multi-Site contests.
- PC² copies the input data file for a problem into memory each time a team program for that problem is executed (this is how the architecture manages the insertion of the input data into the input stream of the team program). If the size of the input data set (file) for a problem is particularly large and the system has a relatively small amount of memory, it is possible to exceed the memory limits of the Java Virtual Machine (JVM). This problem can be circumvented by utilizing a script for the “Program Execution Command Line” which copies the required data file into the **execute**¹⁶ directory and then invokes the team program. (See the following section on Contest Languages, and the Appendix on Language Definitions, for further details.)

6.4.2 Problem Dialog controls

- Show the output window – shows the PC2 output window upon completion of execution/validations
- Show Compare – shows the PC2 compare window upon completion of validation
- Hide Problem – do not show this problem to the teams, will not show this problem on the Scoreboard HTML/output

6.4.3 Defining a Problem Validator

See Appendix E - Validator

6.4.4 Defining Automated (Computer) Judging

By default each problem is manually judged, a person selects a judgment for each submitted run. The system can also automatically judge runs with an optional second step where a person/judge manually judges the run.

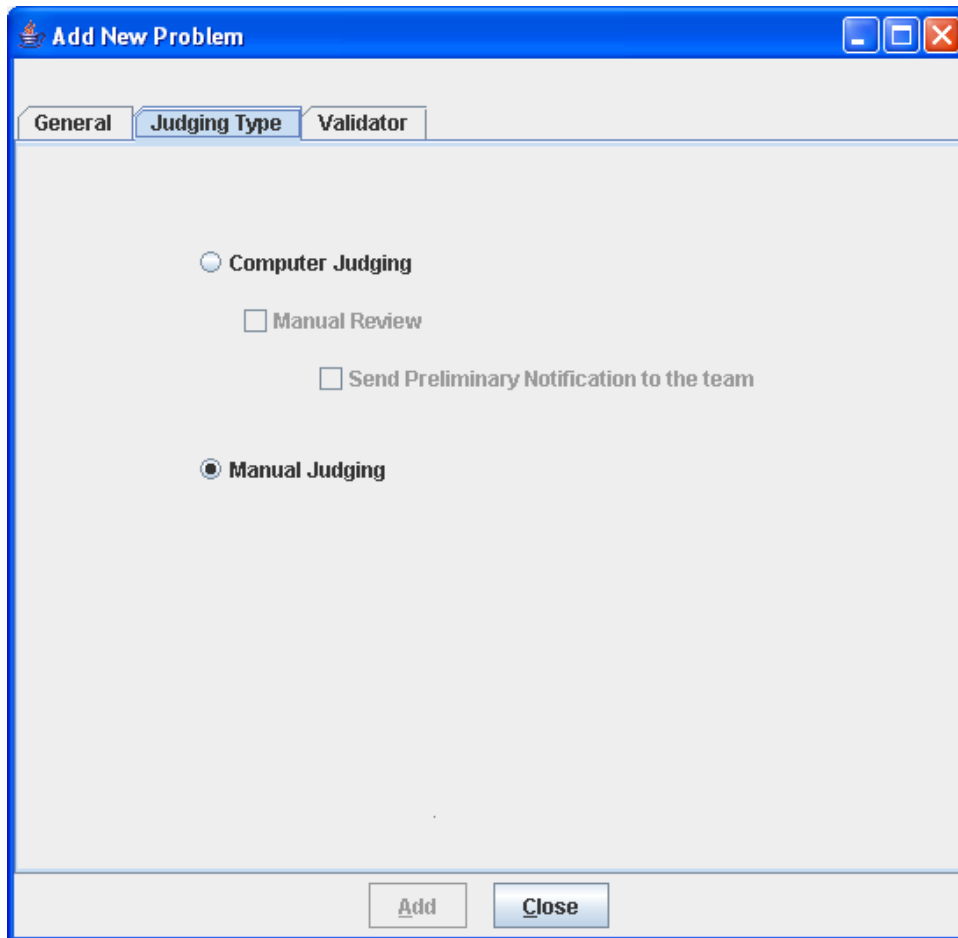
For the system to automatically judge a problem a validator must be defined for that problem, the Judging Type settings needs to be set as Computer Judging and at least one judge module must be configured to judge the problem.

¹⁶ Note that execute directory is name executesite1judge1 for judge 1 from site 1.

6.4.5 Adding a Validator to a Problem

See Appendix E for instructions about how to define a validator for a problem.

6.4.6 Changing problem to Computer Judging



To define a problem to be computer judged, perform the following steps using the **Judging Type Tab** on the **Add New Problem** dialog:

➤ Select Computer Judging

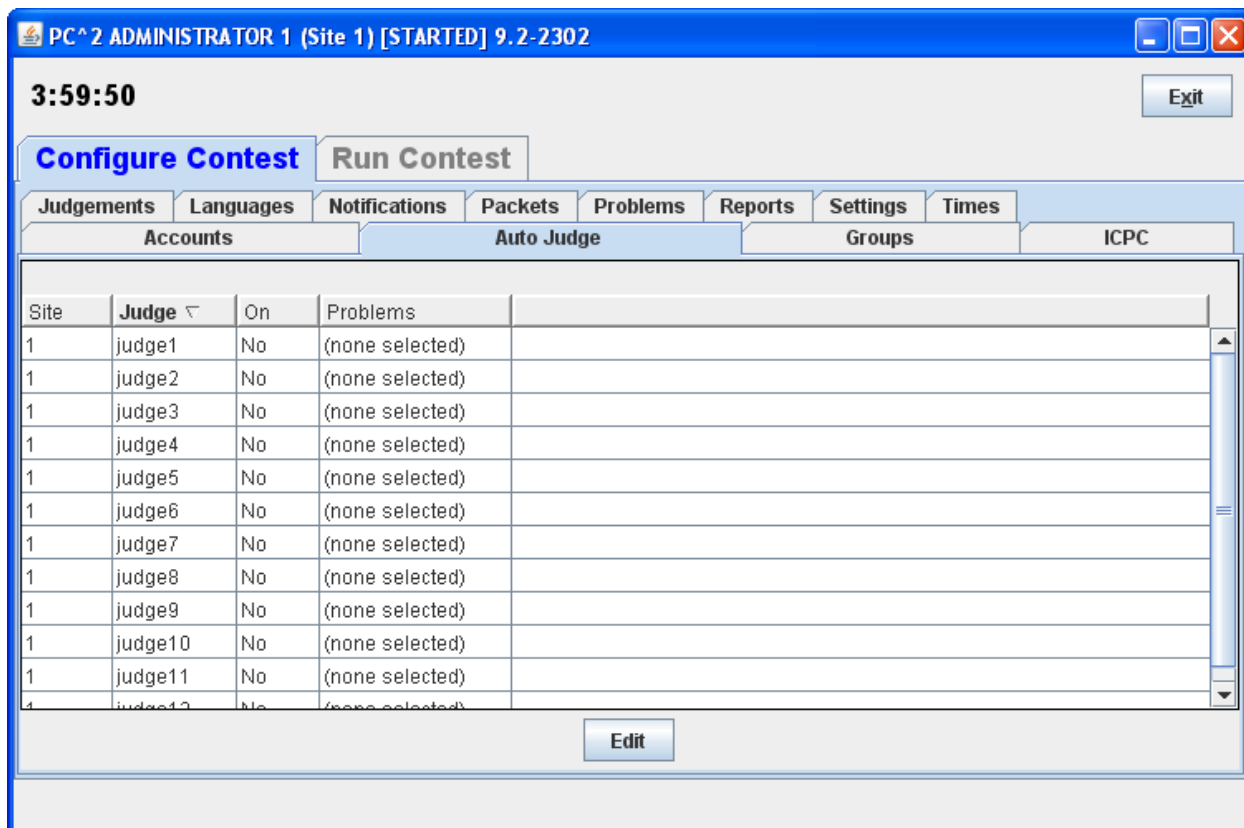
1. Optionally select Manual Review
 - a. Optionally select Send Preliminary Notifications to teams¹⁷

¹⁷ If the Send Preliminary Notifications to teams is not selected the automatic computer judgement will not be shown to the team.

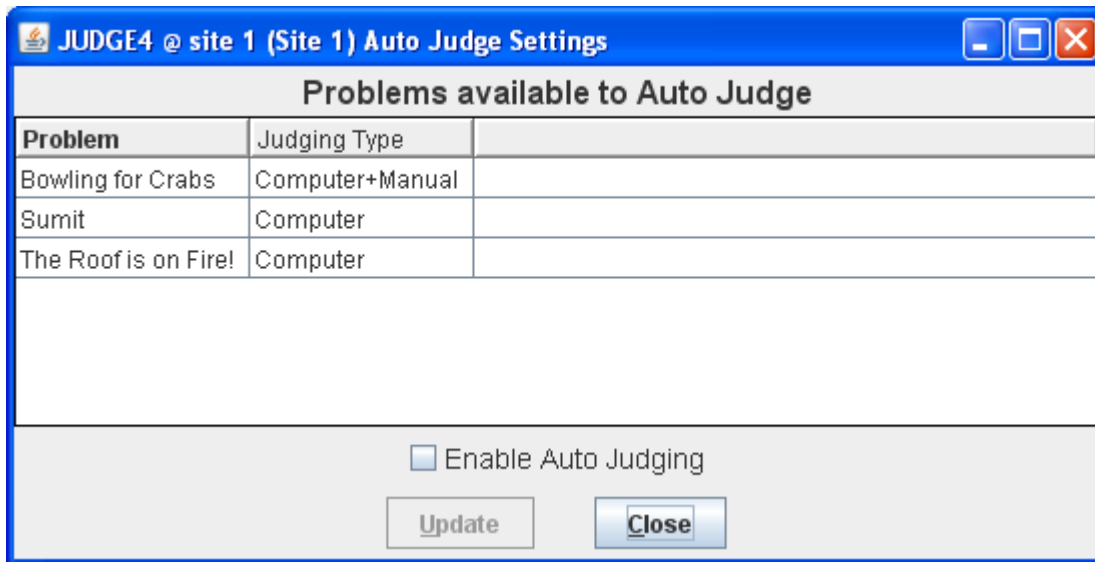
If a validator is not defined and the Judging Type for a problem is Computer Judged the system will not be allowed to be saved and a message “Computer Judging selected, must select a validator”. Once the validator is defined/selected then the problem definition can be saved.

6.4.7 Assigning Auto Judging to Judge module

After Judge login accounts have been created, problems that are computer judged can be assigned to judge modules. Use the Auto Judge Tab on the Configure Contest Tab on the Administrator module.

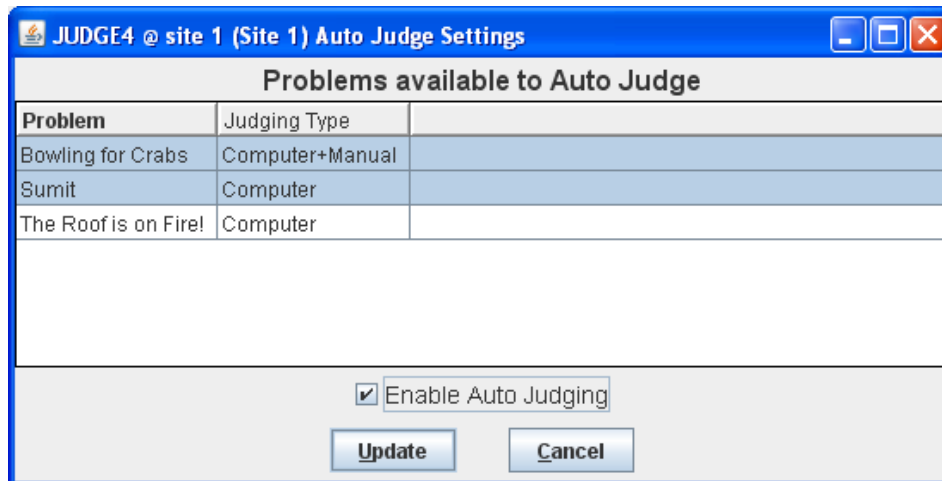


Select a judge account and click Edit. The Edit dialog will only show problems that can be automatically judged (have a validator defined).

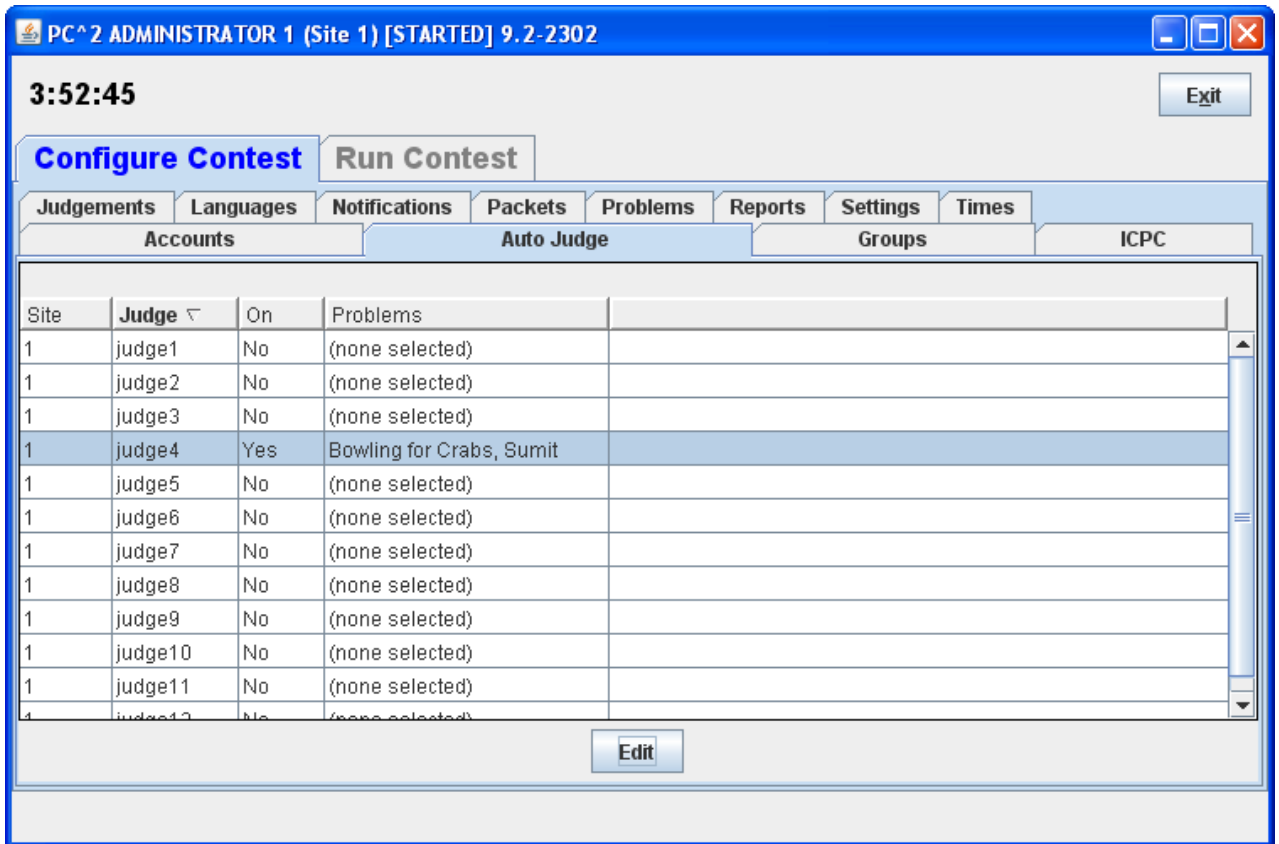


Select the problems that this judge, in this example Judge 4 site 1, will automatically judge. Use [Ctrl] and Click each problem to select or un-select a problem.

To start auto judging (now) check Enable Auto Judging and click Update.



Upon clicking update the Auto Judge Tab will change to the following:



the Auto Judge Tab shows that Judge 4 has computer judging ON and will automatically select and judge two problem.

6.4.8 Starting Auto Judging

When a judge account is logged in then computer judging will start. Runs will be automatically selected in chronological order and judged. To monitor the status of runs use the Runs Tab under the Run Contest Tab.

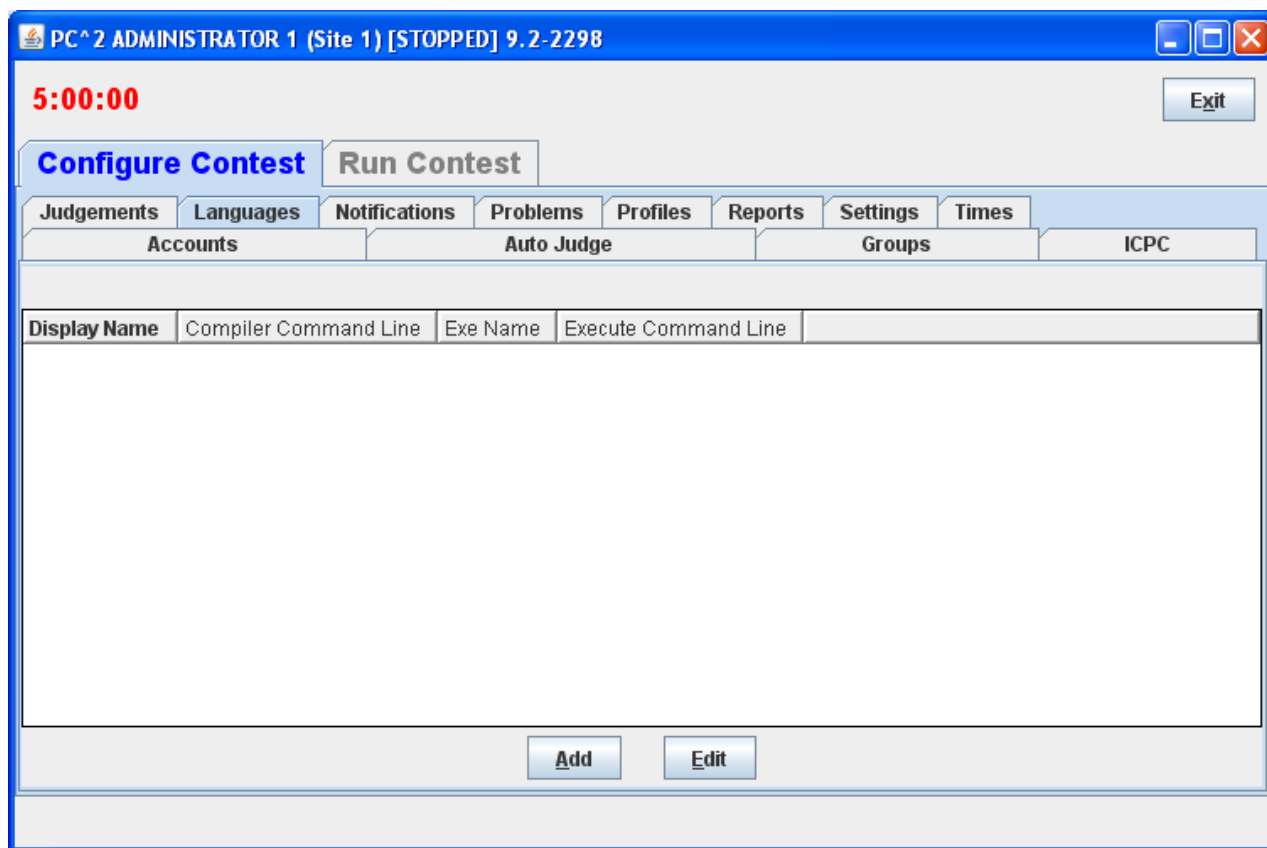
6.4.9 Stopping Auto Judging

To stop a judge module from judging, Edit the Auto Judge (Settings) and uncheck the Enable Auto Judging Tab.

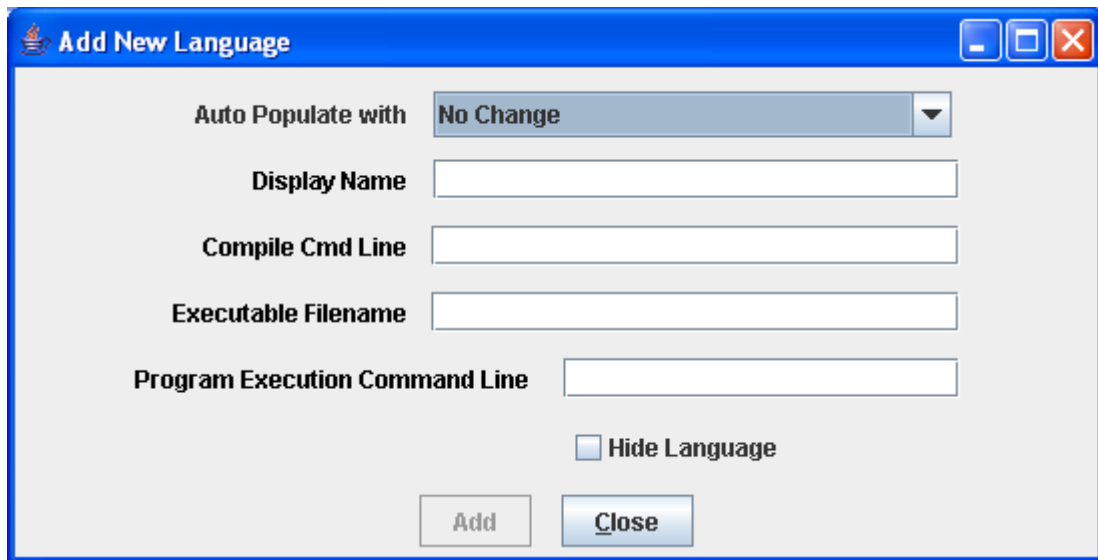
6.5 Contest Languages

6.5.1 Defining a Language

PC² must be provided with information about the programming languages used by contestants (Teams). To enter this information, click the **Languages** tab on main Administrator screen. This will bring up a display similar to the following:



Note that the display is empty because no languages have been defined yet. To add a language description, click the **Add** button. This will bring up an **Edit Language** dialog, similar to the one shown below, containing four fields used to describe the language to PC².



The “Display Name” for a language is the name which Teams will see when they are asked to specify the language in which they have written a program which they are submitting. The Display Name can be any arbitrary text; it does not have to be a real language name (for example, “Local C Compiler” could be a legitimate language Display Name).

The “Compile Cmd Line” field is used to specify the command line which is used to compile source code and produce an “executable program file” in the language.

The “Executable Filename” field is used to tell PC² the name (or more correctly, the form of the name) of the output (executable program) file produced by the compilation process. PC² clears its internal execution directory of any instance of the specified executable file prior to compilation, and checks for the existence of the specified executable file following compilation. It interprets the existence of a new executable file as evidence of successful compilation.

The “Program Execution Command Line” field is used to specify the (form of the) command line required to execute (run) the resulting program. Execution is only performed if the preceding steps were successful in producing a new executable file.

The Contest Administrator must define each language to be used in the contest by filling in the four language definition fields, replacing the default values shown above with the appropriate values for the language being defined. (Note that the preceding example screen shows values called “command parameter substitutions” in the language definition fields; see the following sections for further details on the definition fields.)

Once the definitions for a language have been entered, click the **Update** button to store the information and return to the main Administrator screen. The language names will be displayed under the **Languages** tab on the main Administrator screen, as shown below. To add more languages, click the **Add** button again to return to the **Edit Languages** screen. To modify a previously-entered language, click on the row containing the language description to select it and then click the **Edit** button. See the Appendix on Language Definitions for further details.

6.5.2 Command Parameter Substitutions

The four language description fields in the **Edit Language** dialog can be “hard-coded” by entering fixed values if desired. For example, the Display Name for a language is normally fixed for the duration of a contest (e.g., “Java”, or “C++”, or “Pascal”).

However, entering fixed values for the Compile Command, Executable Filename, and Program Execution Command fields can be extremely cumbersome and inflexible – the details of these fields may need to change with each different program file submission, for example. In order to provide more flexibility, PC² supports the use of “parameter substitutions” in these fields.

PC² parameter substitution fields are indicated by matching curly braces, with the first character inside the left curly brace being a colon (‘:’). Following the colon character is exactly one of the predefined PC² parameter substitution keywords. Any number of command parameter substitution fields may appear anywhere in a language description field. The currently defined parameter substitution keywords and their corresponding meanings are given below.

Keyword	Meaning
mainfile	Replace with the full name of the submitted file, including any extension (but excluding any ‘path’ specifier on the front of the filename)
basename	Replace with the base component of the file name, omitting any extension (and excluding any ‘path’ specifier on the front of the filename)

The following section shows examples of language definitions, including the use of command parameter substitution fields.

6.5.3 Language Definition Examples

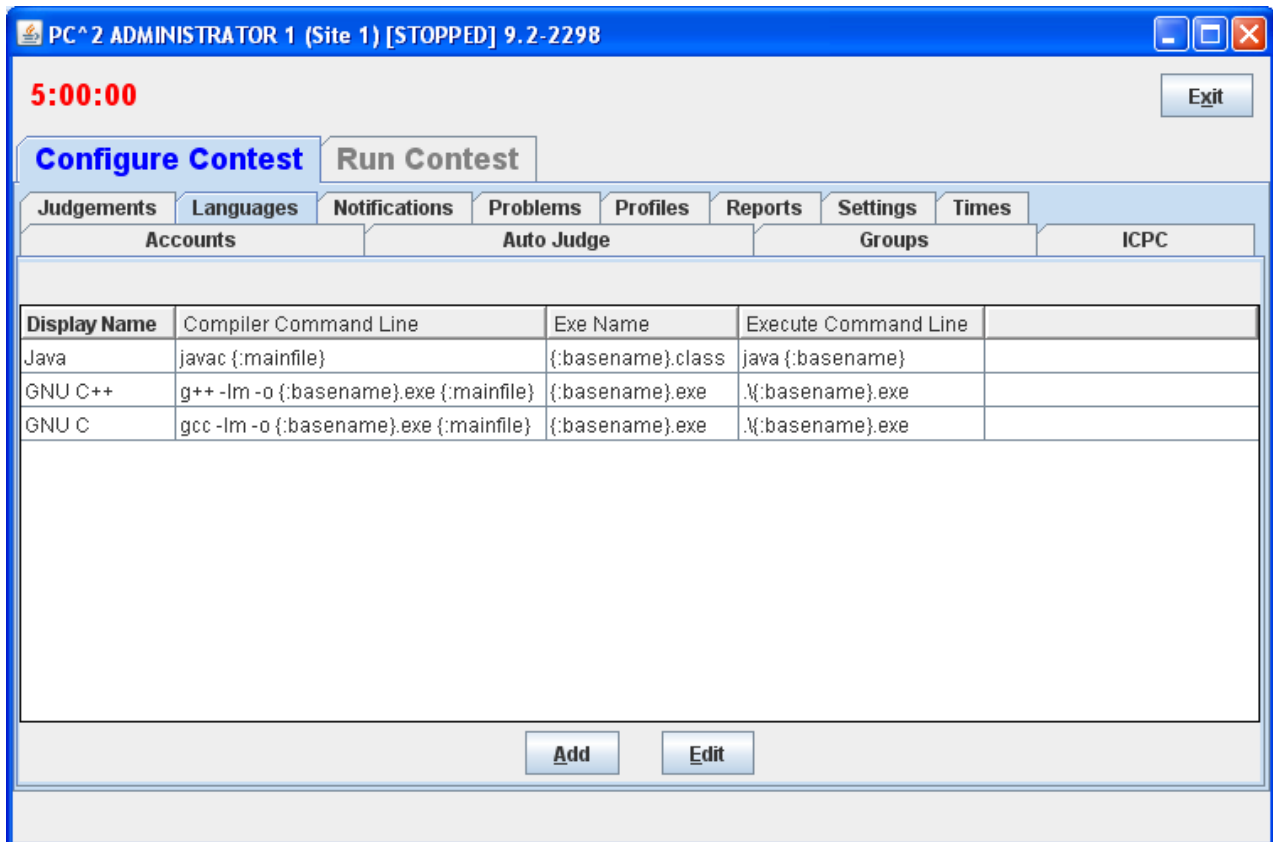
The following screen shows a set of filled-in fields defining a language named “C” and using the GNU GCC compiler.

The compile command line invokes the compiler (“gcc”) and passes it an argument specifying use of the math library (“-lm”). The compile command line also specifies the assignment of a specific name to the “object” (compiled output) file (the “-o” argument), followed by the name to be assigned to the object output file. In this case, the object output file is to have the same name as the base name of the input source code file. (So for example if a team submitted a file named “proga.c”, the object output file would be named simply “proga”, since that is the value to which the “{:basename}” substitution parameter would be expanded.)

The final argument on the compile command line gives the name of the source file to be compiled, which would be expanded from “{:mainfile}” to become “proga.c” if that was the name of the submitted main program source file.

The Executable Filename field indicates that the executable file which produced by the compile command has the same name as the base name of the submitted program; this is because the compile command specifies (via the “-o” argument) that this is the executable file name which should be produced.

The Program Execution command field specifies that the command used to execute the compiled program is simply the same as the name of the executable file produced by the compilation step (and specified in the Executable Filename field), which in this case is again the base name of the original source code file.



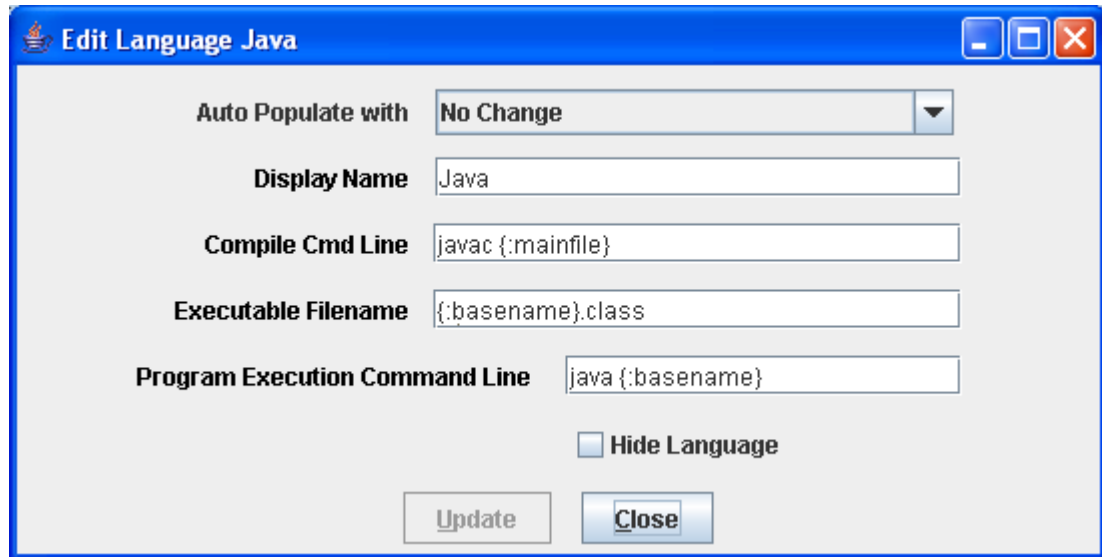
If a team were to submit a C program in a file named **proga.c** using the above language, PC² would first execute:

```
gcc -lm -o proga proga.c
```

to compile the program (substituting **proga** for the {:basename} parameter and **proga.c** for the {:mainfile} parameter). It would then check for the existence of an executable file named **proga**, and if that file exists then PC² would request the underlying operating system to execute the command:

```
proga
```

The following screen shows a second language definition example: a definition for a language with the display name “java”:



If a team were to submit a Java program in a file named `sumit.java` using this language definition, then PC² would execute the following command to compile the program:

```
javac sumit.java
```

Then PC² would check for the existence of an executable file named `sumit.class` and if that file exists then PC² would execute the following command to run the program:

```
java sumit
```

Note that the form of the language definition fields differs somewhat between the previous example (C) and the second example (Java). This is because of the different ways in which these two languages define the compilation and execution process. Notice also, however, that while the language paradigms are different, the use of command parameter substitutions allows the Contest Administrator easily to provide descriptions of how to handle the differences. The appendices contain further samples of language definitions for specific compilers.

6.5.4 Language Definitions In Multi-Site Contests

Language definitions in PC² are *global*. This means that, just as with Contest Problem definitions, when a language definition is entered at one site in a multi-site contest, that language definition will be visible at *all* connected contest sites. However, unlike the situation with Contest Problems (where the problem definitions are usually identical across sites), language definitions may differ between sites – even for the “same language”.

For example, it may be the case that every site allows the use of the “C” language. However, it may also be true that the specific command sequence to invoke the C compiler may differ between sites: a different C compiler might be used at different sites, or even if the same compiler is used it may be necessary to allow for differences in the “path” needed to access the compiler or for other environmental differences.

One way to deal with differences in language details between sites is to create a different PC² language description for each different language/site combination. This can quickly become cumbersome, however; for example, if there are four languages (e.g. C, Java, Pascal, and Perl) and five sites using those languages, it could require entry of up to 20 different language descriptions (Site1C, Site2C... Site1Java, Site2Java,... etc.). This can become particularly unwieldy for Teams, who must search through a list of 20 different languages looking for not just the correct language but the correct language *for their site*.

To avoid this combinatorial explosion of language definitions, a simple technique can be used when defining languages in a multi-site contest: use of *generic language scripts*, tailored at each site for the site-specific configuration.

For example, consider a contest using, say, C, Java, and Pascal. The Contest Administrator should define those three languages in PC² using the actual language names (“C”, “Java”, and “Pascal”) as the PC² “language Display Names”. However, rather than defining a specific compilation command for each language (which may differ between sites), each language should have as its compilation command a command which invokes a language-specific (but site-independent) *script* (or “batch file”) designed to compile a program in that language.

In other words, for the above three languages, PC² language definitions would be created to define the “compilation command” for the language named “C” to be the invocation of a script (batch file) named “*compileC*” (or “*compileC.bat*”); the compilation command for Java would be the invocation of a script named “*compileJava*”; and the compilation command for Pascal would be the invocation of a script named “*compilePascal*”.

Then, at *each site*, the Site Director is responsible for placing on machines at that site a set of scripts or batch files of the corresponding names (e.g. *compileC*, *compileJava*, and *compilePascal*). Within each script at each site is a set of *site-specific commands* which perform the necessary steps (compile a C program, compile a Java program, or compile a Pascal program) in the appropriate site-specific manner.

Note that if necessary, the same technique of “generic scripts” which vary between sites can also be used in specifying the details of “Program Execution Command Line” for languages. That is, the Contest Administrator can specify “*executeC*”, “*executeJava*”, and “*executePascal*”

scripts for the program execution language definitions in PC² and then arrange for appropriately different script contents at each site.

Note also that PC² “command parameter substitutions” may be used in compilation and execution command lines independently of whether the command is invoking a script or not; in this way the Contest Administrator can arrange to pass necessary data (such as the main program file name and/or the base name) to a script.

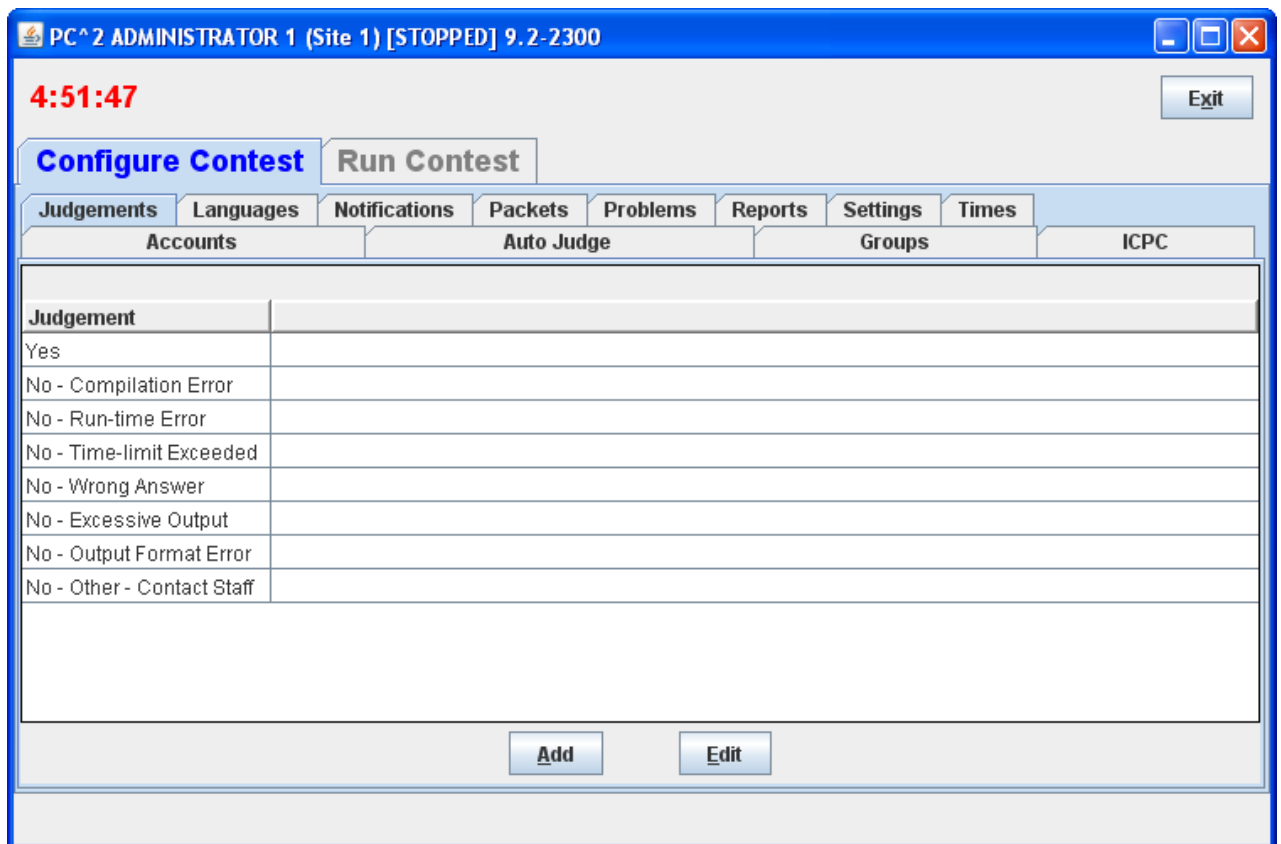
Using generic script names in PC² language definitions and providing site-specific implementations of each language script at each site allows the Contest Administrator to significantly reduce the number of language definitions which teams must deal with, while at the same time retaining the flexibility necessary for dealing with site differences in a multi-site contest.

6.6 Contest Judgments

6.6.1 Defining a Judgments

On the Administrator module the contest judgment messages can be viewed, added, edited and deleted. The Judgments Tab under the Configure Contest tab on the Administrator shows all the judgments available.

By default a list of judgments is loaded as seen in the Judgments Tab below.



To add a judgment click the **Add** button. This will bring up an **Add New Judgement** dialog, similar to the one shown below, containing two fields used to define a judgment.



The “Judgement” for a judgment is the name which Judges will see when they are asked to judge a run. This name is also seen by the Team’s when they receive that judgment.

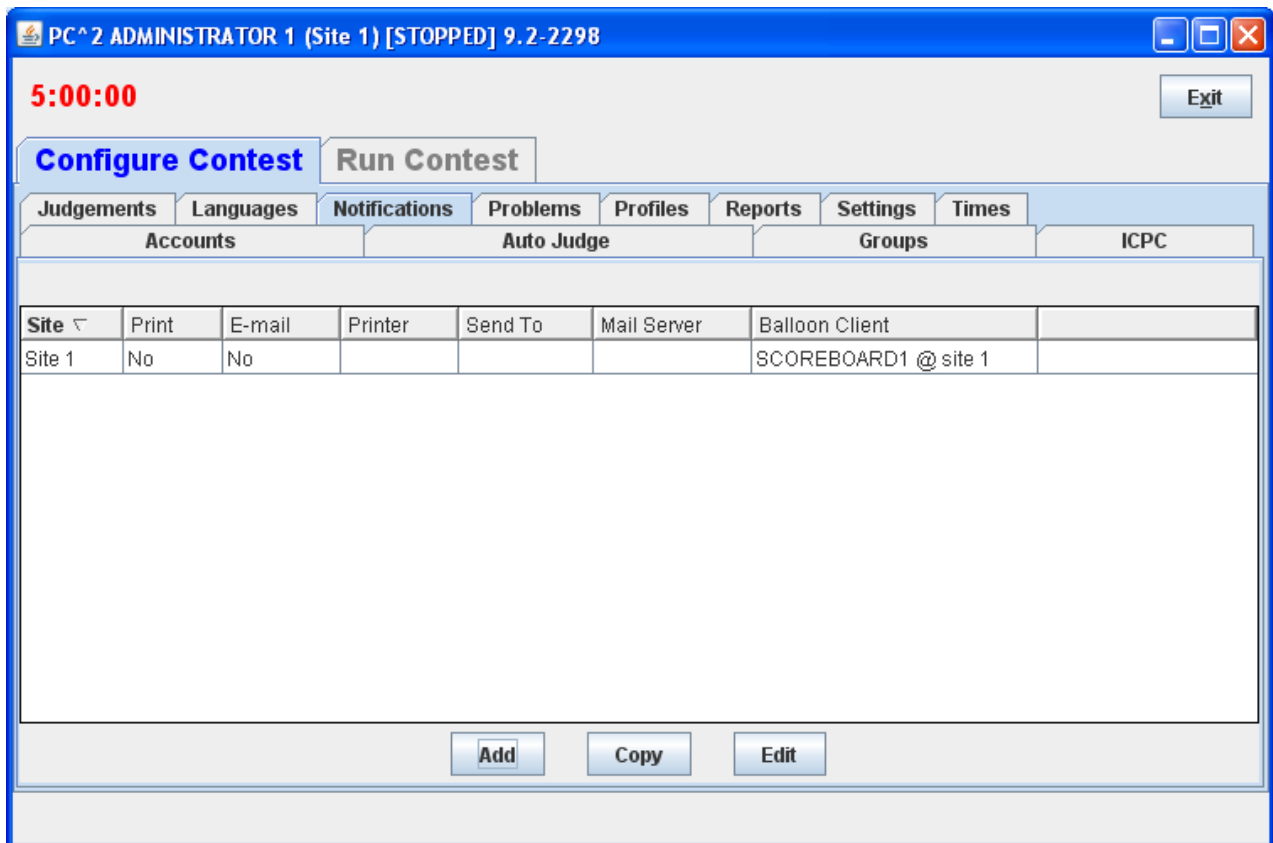
The “Hide Judgement” will remove (hide) this judgment from the list of judgments that the Judges can use.

To change an existing judgment click the **Edit** button. This will bring up an **Edit Judgement** dialog, similar to **Add New Judgement** dialog.

6.7 Notifications (Balloons)

In many contests (including the ICPC World Finals), balloons are used to indicate to contestants and spectators alike the general state of the contest. Each time a team solves a problem, a balloon of a specific color is sent to the team and attached on or near their machine. As the contest progresses, the contest floor gradually fills up with a multi-colored display showing how various teams are doing in the contest. (This is a colorful and normally well-received operation; if you have never tried it, we recommend doing so.)

Using balloon notifications in a contest does present some additional management overhead (keeping track of which team should get what color balloon, etc.). Since PC² was designed to support the ICPC World Finals (as well as its Regional and Local contests), it contains some built-in support for “balloon operations”. Selecting the **Options** tab on the main Administrator screen and then clicking on “**Balloon Options**” will cause a dialog similar to the one shown below to appear; this dialog is used to configure the handling of balloons in a contest.



6.7.1 Defining a Notification

Balloon options include ability to specify the color of balloon associated with each problem; sending messages to a printer each time a balloon should be delivered to a team; and sending an email message via a specified email (SMTP) server to an arbitrary email account each time a balloon should be delivered to a team. Printed and emailed messages contain the relevant details such as Team, problem, and balloon color. Each of these options can be selected on a per-site basis (so that, for example, sites can use different color balloons for a given problem).

Problem	Color
Hello	
Sumit	
Bowling for Crabs	
Problem Solved	

Generation of balloon notifications is handled by the PC² Scoreboard machine.¹⁸ Once email and/or printing notification is enabled, every “YES” judgment detected by the PC² Scoreboard “board1” account will cause an email notification and/or a printed notification to be sent to the configured location.

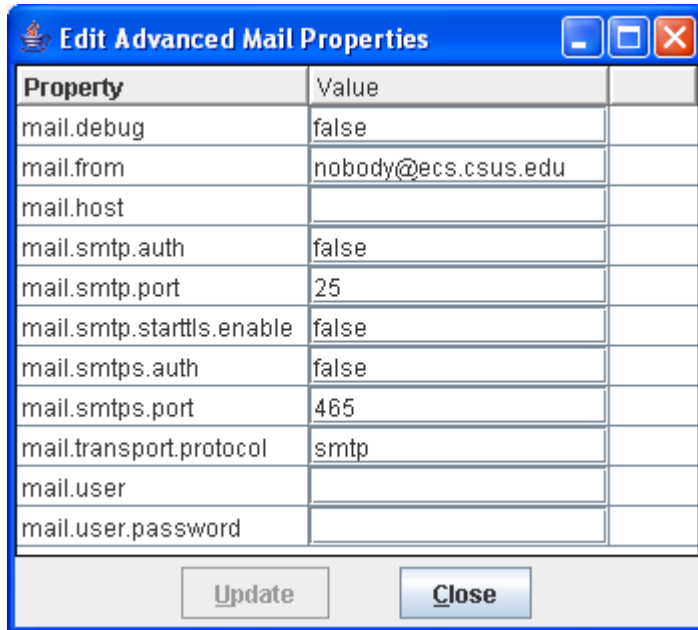
To enable the use of email balloon notifications, check the “Enable Emailing” box, then enter in the appropriate text boxes the full name of an SMTP email server accessible to the PC² Scoreboard machine along with a valid email address (EMail contact).

To enable the use of printed balloon notifications, check the “Enable Printing” box, then enter in the appropriate textbox the device identifier of a printer accessible to the PC² Scoreboard machine. If it is also desired to print notifications of “NO” judgments, check the “Enable Sending No’s” box.

¹⁸ More specifically, it is handled by a PC² Scoreboard machine logged in under the PC² account selected as the “Balloon Client”.

6.7.2 Email Server Advanced Settings

If there are non-standard SMTP or additional SMTP settings required the Email Server Advanced Settings can be used to specify the settings.



Property	Value
mail.debug	false
mail.from	nobody@ecs.csus.edu
mail.host	
mail.smtp.auth	false
mail.smtp.port	25
mail.smtp.starttls.enable	false
mail.smtps.auth	false
mail.smtps.port	465
mail.transport.protocol	smtp
mail.user	
mail.user.password	

SMTP Authentication is now supported. SMTP Authentication may require some additional setup to support smtps (SSL), if there are questions about how to use SMTP Authentication send email to the PC² team (pc2@ecs.csus.edu).

6.8 Contest Profiles

This feature is scheduled to be released in a future version of PC².

The Profile tab provides a facility for describing different contest configurations – for example, the differences between a *Practice Contest* and a *Real Contest* which follows the Practice Contest and uses the same set of account information (Teams, Judges, etc.) and language definitions but uses a different *problem set*.

6.9 Options (Settings tab)

The **Settings** tab on the main Administrator Configure Tab displays a screen which allows selection of additional options which can be used to display and/or control various aspects of a contest.

PC^2 ADMINISTRATOR 1 (Site 1) [STOPPED] 9.2-2302

5:00:00

Exit

Configure Contest Run Contest

Judgements Languages Notifications Packets Problems Reports Settings Times

Accounts Auto Judge Groups ICPC

Contest Title Programming Contest

Team Information Displayed to Judges

None Show Numbers Only Show Names only

Show Number and Name Show Alias

Maximum output size (in kB) 512 Edit Scoring Properties

Judges' Default Answer No response, read problem statement

Update Cancel

Contest Title - Specifies the contest title which is to be displayed on the scoreboard.

Team Information Displayed to Judges - Specifies whether or not to reveal the identity of Teams to the Judges while a run is being judged. In the examples below team 5 has a display name of “CSUS Hornets” and an alias of “Team Orange”

- None – show *** for team name
- Show Numbers only – show team# for team name, ex team5
- Show Names only – show display name only, ex CSUS Hornets
- Show Number and Name, ex 5 CSUS Hornets
- Show Alias¹⁹ – show an alias for the team name, ex Team Orange

¹⁹ To load/specify team aliases see the section Loading Account Data

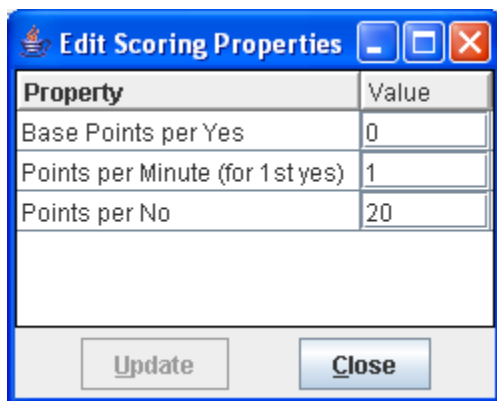
Maximum Output Size - Specifies the maximum amount of output, in Kbytes, which a team program is allowed produce to stdout or stderr. Any output beyond this amount is discarded by the system and a message is added to the end of the output. The default value is 512 K (1/2 MB).

Edit Scoring Properties – specifies the scoring point penalties, see section Edit Scoring Properties.

Judges' Default Answer – specified the clarification answer sent to teams if judge selects the **Default Answer** button while answering a clarification.

6.9.1 Edit Scoring Properties

The Edit Scoring Properties provides a way to modify the PC² Scoring Algorithm; this algorithm is used to determine the contest standings. These values add time to the total time for a team on the standings.



Base Points per Yes – for the first Yes/Correct solution per problem add this number of points to the Team's Time.

Points per Minutes (for 1st yes) – for the first Yes/Correct solution per problem multiply this value with the elapsed time minutes and add the result to the Team's Time.

Points per No – for each solved problem, multiply this value by the number of No/Incorrect runs before the first Yes.

Note that changing scoring property values only affects that particular scoring criterion; it does not alter the overall determination of ranking criteria. For example, assigning a negative value to one property does not change the fact that teams are ranked first by number of problems solved; a team with a large negative score will still not be ranked higher than a team which has solved more problems.

6.10 Sites

This tab on the Administrator main screen displays a list of all the sites in the contest; this list should be checked to verify that PC² knows about all sites. If the site is currently active (connected to the rest of the contest) the IP address for that site's server is displayed.

PC² ADMINISTRATOR 1 (Site 1) [STOPPED] 9.2-2302

5:00:00 Exit

Configure Contest **Run Contest**

Plugin Load Reports Runs **Sites** Standings Standings HTML Team Status
Connections Clarifications Finalize Logins Messages Options Packets

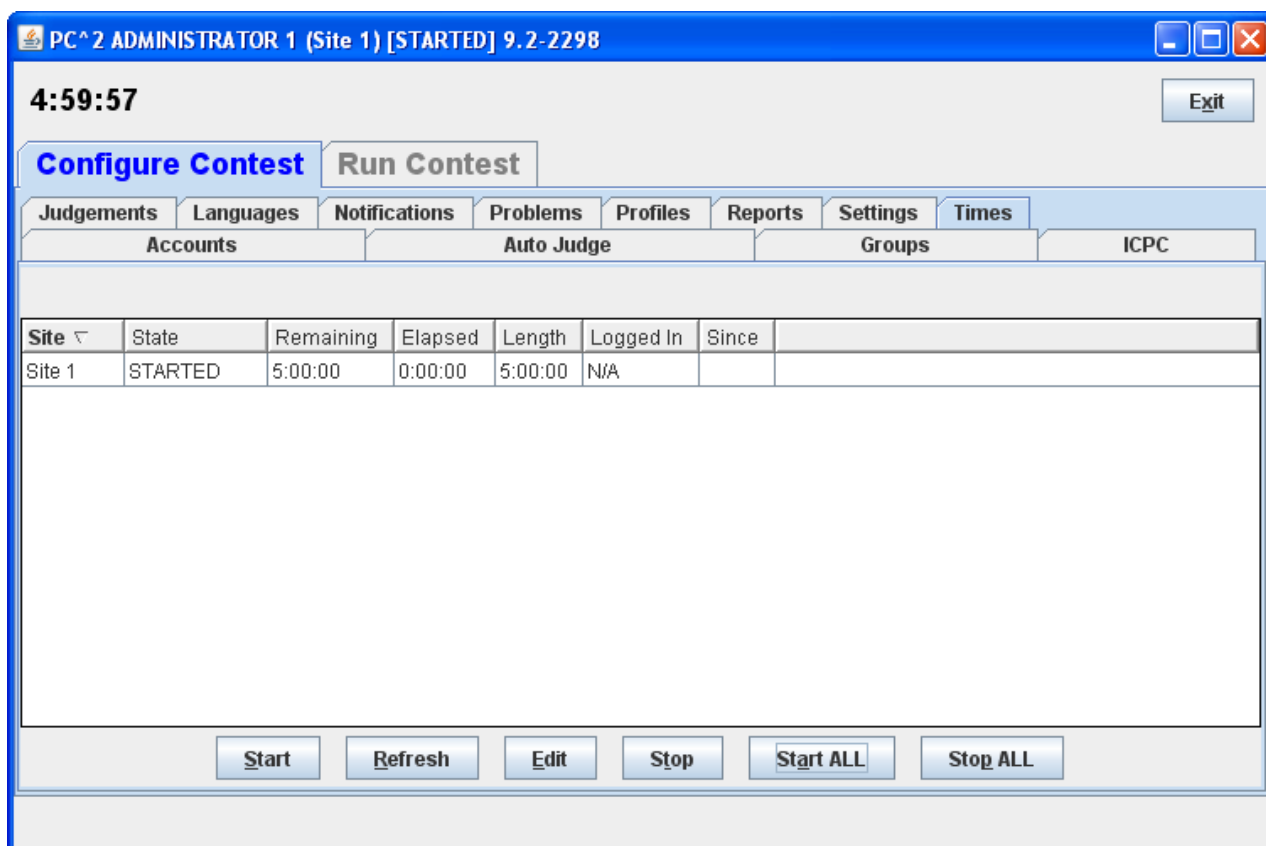
Site Number	Site Title	Password	IP	Port
1	Site 1	site1	localhost	50002
2	Site 2	site2		50002

Add Site Update Site Cancel Reconnect Shutdown

7 Starting the Contest

7.1 Clock Control

Once the contest has been fully configured in PC², the contest clock must be started before teams can submit runs (the Team client will not allow run submissions if the contest clock has not been started). The **Times** tab on the Configure Contest tab on the Administrator screen is used to control the contest clock display and to start and stop the contest clock. Clicking the **Times** tab produces a screen similar to the one shown below:



The **Start** button is used to tell PC² to start the contest clock for the selected site. Pressing **Start** starts the contest clock running for that site and allows teams at that site to submit runs. Pressing the **Start ALL** starts the contest clock at all sites.²⁰

The amount of time remaining in the contest at the current site is displayed in the top left part of the window just below the window title, the example above shows 4 hours 51 minutes and 45 seconds. The remaining time automatically starts counting down as soon as the **Start** button is

²⁰ If the contest clock for another site is already started, then this command causes no harm.

pressed. It continues to automatically update (count down) as long as the contest clock is running.²¹

The remaining and elapsed time in the grid will not update automatically, to update those times use the **Refresh** button

It is important to note that, from the point of view of PC², the contest does not start until the **Start** button is pressed. For this reason **it is important that the Contest Administrator remember to press the Start button at the actual time the contest starts.** Failure to do this can produce erroneous scoring results.

Typically, for example, a contest is deemed to have “started” when the contest problems are distributed to the teams. If the PC² **Start** button is not pressed for another, say, 15 minutes, then that 15 minutes will not be considered to have been part of the contest by PC². If a team were to submit a run 20 minutes after the contest started (i.e. 20 minutes after the problems were handed out), the timestamp on that run would show a contest elapsed time of 5 minutes, not the correct value of 20 minutes. This would produce erroneous values on the scoreboard.

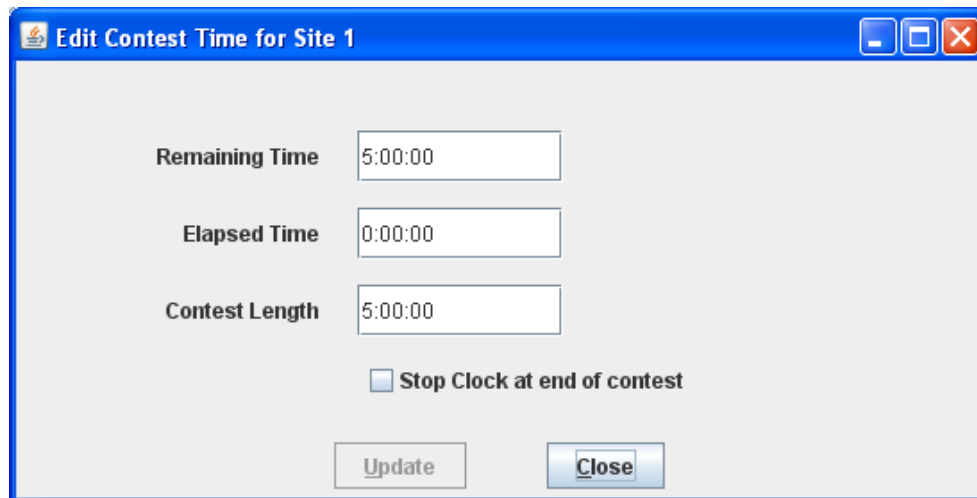
The **Stop** button is used to tell PC² to stop the contest clock for the selected site. The **Stop** button can be used to insert a pause in a contest (for example, to allow a break for lunch). During the time the contest is stopped, the contest clock at the site does not count down, and teams are prohibited from submitting runs. Also when stopped the contest clock displays in **RED**. When the **Start** button is pushed again, the contest clock for the site picks up where it left off.

Note that this means that if a team submits a run one minute before the contest clock is stopped, and then the clock is stopped for 30 minutes of real time, and then the team submits another run immediately after the contest clock is restarted, the timestamps on the runs will be one minute apart. In other words, PC² does not consider time during which the contest clock is stopped to be part of the contest. (If this is undesirable – that is, if the Contest Administrator wishes *all* time which elapses to be counted, then simply do not press the “stop” button once the contest has been started.)

7.2 Contest Length

The **Edit** button on the Time tab displays the **Edit Contest Time** dialog (shown below) which allows the Administrator to change the elapsed, remaining or contest length.

²¹ The remaining time is also displayed on the Team and Judge client screens, and counts down automatically; however, it is displayed only to a resolution of one minute on those screens. When the Team or Judge window is minimized the remaining time countdown is displayed in the title.



The time values in the **Contest Time** dialog do *not* update automatically when the contest is running; they display only the instantaneous time values at the moment the dialog is activated. If the contest is stopped when the dialog is activated, those times remain accurate indefinitely.

If a new Contest Length which is less than the current Elapsed Time is entered, the system displays a warning message to inform you that setting the contest length to a value less than the elapsed time will effectively mean the contest is over.

The “**Stop clock at end of contest**” checkbox has no effect. Bug 378 has been filed to make this functionality work.

7.3 Multi-Site Clock Control

As described above, the contest clock in PC² operates on a per-site basis. That is, each site in a multi-site contest has its own contest clock, and PC² keeps track of “contest time” independently at each site. This is done to allow support for independent time-management constraints at different sites, and allows scoring to be done accurately without worrying about differences in timing between sites (e.g., a necessary pause at one site which does not affect other sites).

Each PC² site server determines the time of submission of a run from a team in terms of “contest elapsed time”, which means that a submission will be marked (“time-stamped”) according to the contest elapsed time at that site. The scoreboard in turn computes rankings based on this “submission time”, which means that overall (multi-site) rankings will be determined according to “contest elapsed time” at the site from which each run originated. This method puts teams at all sites on an equal competitive footing regardless of differences in the time at which the contest actually starts at each site.

However, this mechanism (keeping track of contest time independently at each site) can produce erroneous scoring results if the Contest Administrator does not take care to control the multi-site contest clocks correctly. Specifically, in a multi-site contest **it is important that the clock at each site be started at the moment the contest starts at that site.**

Typically, for example, a contest is deemed to have started at a site at the moment the contest problems are distributed to teams at that site. If this event (problem distribution; contest start) happens at different real times at different sites, then a Contest Administrator at each site should press the **Start** button at that site precisely when the contest starts *at that site* – regardless of whether the contest has started simultaneously at other sites. In this way, runs submitted by teams at each site will be correctly time-stamped with the true “contest elapsed time” as their “submission time”.

If all sites in the contest are fully connected, and the contest problems are handed out at the same instant at all sites, then a *single* Contest Administrator can easily coordinate the start of “contest time” at all sites correctly, simply by using the “**Start All Sites**” button on the **Time Tab**. This button performs the same function as **Start** except that it applies the corresponding action (starting the contest clock running) simultaneously to all connected sites. (Use the Admin **Sites** tab to determine which sites are connected; connected sites are those that have valid IP addresses displayed in the **Sites** screen).²²

In a fully-connected multi-site contest where Contest Administrators at each site have agreed (e.g. by telephone or other method) on the precise instant at which all teams at all sites will receive the contest problems and thus the time at which the contest officially starts, having a single Contest Administrator press the **Start All Sites** button is the preferred (safest) way to coordinate the start of a contest. Likewise, if all sites are tightly coordinated, the **Stop All Sites** buttons can be used to stop the contest clock simultaneously at all connected sites.

However, if there are some sites which do not have network connectivity, or some sites at which the distribution of the contest problems (and hence the start of the contest) is delayed for some reason, it is *critical* that a Contest Administrator *at that site* makes *certain* that the contest clock is started *at that site* at the moment the problems are handed out (or whatever other criteria determine the moment in time when the contest starts at that site).

We have seen more than one instance of a situation in a multi-site contest where the contest problems were handed out at all sites (hence, the contest is effectively under way at all sites), but one or more sites failed to notify PC² that the contest had started (either because the sites were not networked, or because the **Start All Sites** button was not used). In this case, if say 30 minutes elapsed before PC² is notified to start its contest clock at one site, then teams at that site will effectively get 30 “free” minutes – a run submitted 31 minutes after the problems were handed out will appear to PC² at that site to have been submitted “1 minute into the contest”.

Caveat Administrator:

It is *critical* that the PC² “contest clock” be started, at every site, at the time the contest starts *at that site*.

In addition, recall that as noted previously the **Stop clock at end of contest** checkbox is entirely *local* in function. This box must be separately checked on the Admin client for each site

²² If a new site server is started after the **Start All Sites** button is pressed, it will be necessary to use an Admin client to start the contest time for that site. Alternatively, you could use **Start All Sites** to start all sites including the new site

where it is desired that the contest stop automatically when the Remaining Time clock counts down to zero.

7.4 Practice Sessions: Resetting A Contest

In many contests, the overall contest activity starts with a “practice session” prior to the start of the actual contest. The primary objective of the practice session is to ensure that all teams are familiar with the operation of the contest environment (PC² in the present case) prior to the start of the real contest. The practice session might provide teams with a trivial “practice problem” to solve (“print your team name” or “read a file containing integers and print the sum of the integers”, for example), and then require all teams to login to PC² and test out the run submission mechanism by writing and submitting a solution to the practice problem. Some contests also require teams to practice using the PC² “clarification system” during the practice contest. A practice session also has the advantage of giving the Judges practice with how PC² works prior to the start of the real contest.

In order to run such a “practice contest” prior to the start of the real contest, it is necessary to configure PC² for the practice contest. Most of the configuration is identical to what is required for setting up the real contest – creating and configuring accounts, defining languages, etc. The only real difference is typically with the specification of the *problem set*: the practice problem(s) must be configured into the system for the practice contest (it is undesirable to configure the real problems ahead of time, as this would mean the problem names would be visible to the teams during the practice). However, most configuration items other than the problem set are usually exactly the same during a practice contest as they are during the subsequent real contest.

At the end of such practice contest, it is necessary to “reset” the state of the system by removing from the database all runs, clarification requests, judgments, etc. which were submitted during the practice contest. However, it is at the same time desirable to *avoid* removing from the system the “configuration information” such as account names, passwords, language definitions, etc. In addition, it is necessary to switch from the practice problem set to the real contest problem set.

There is a relatively simple way to accomplish a switch between a practice and a real contest while preserving the necessary information. First, create two directories named (for example) **practice** and **real**. Change to the **practice** directory, start a PC² Server in that directory, and then start an Admin and configure a contest including the team accounts, languages, etc., but *omitting the problem definitions*. Next, shut down the server then (recursively) copy the **practice** directory contents to the **real** directory. This creates two directories with identical contest configurations, including accounts, languages, etc., but with no contest problems.

Next, restart a server in the **real** directory and then run an Admin and add the real contest problems to that configuration. Finally, shut down the server, change to the **practice** directory, and restart a server and an Admin in that directory and use the Admin to add the practice contest problem definitions to the **practice** contest configuration. This way, the real and practice contests both have the same accounts, languages, etc., but have different problem sets. To switch to the real contest, shut down the practice contest server, switch to the **real** directory, and start a new server in that directory.

8 Monitoring Contest Status

8.1 Team Startup Status

The **Team Status** tab on the main Administrator screen is used to track the status of Teams once a contest has been started. This is particularly useful during a “practice contest” held just prior to a real contest; it allows the Contest Administrator to verify that all teams have been able to login and use the basic PC² functions successfully.

A sample **Team Status** screen is shown below. Initially all teams are displayed in RED, indicating that the team has not made any contact with the PC² server. When a team logs in, their display changes to YELLOW; when they have submitted at least one run or clarification request their display changes to MAGENTA or BLUE respectively; once a team has successfully submitted both a run and a clarification the display changes to GREEN, indicating the team has successfully performed all the basic PC² functions and should be ready to use the system in the real contest. (The screen below shows teams in each of these states, although it may be hard to read if you are not looking at a color copy of this manual.)

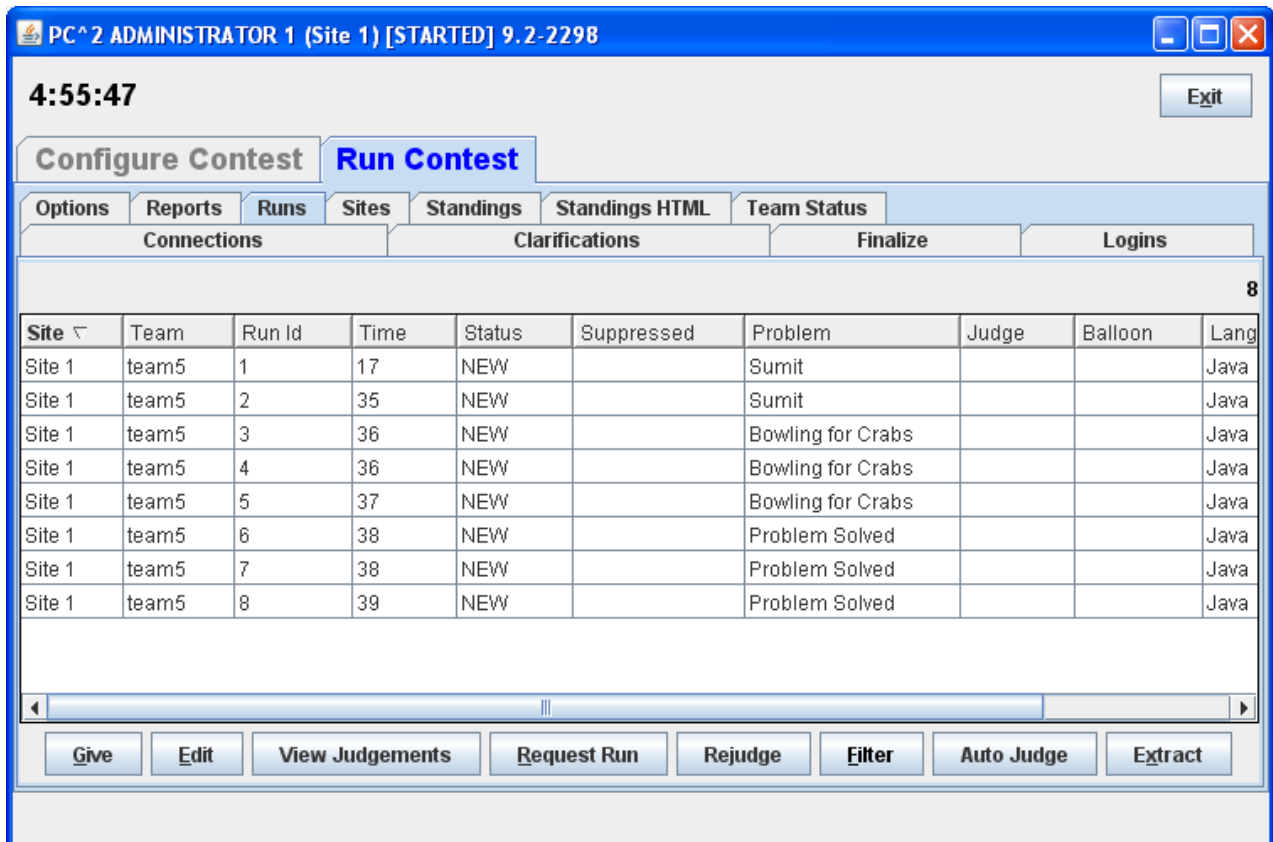
The screenshot shows the PC² Administrator 1 (Site 1) [STARTED] 9.2-2300 interface. The top bar displays the time 4:55:10 and an Exit button. The main navigation includes tabs for Configure Contest, Run Contest, Options, Reports, Runs, Sites, Standings, Standings HTML, and Team Status. The Team Status tab is active, showing a table of team statuses. The table has columns for No Contact, Has Logged In, Submitted Run(s), Submitted Clar(s), and READY. The table shows 22 teams (S1 team1 to S1 team22) with various status indicators.

STATUS							
No Contact	Has Logged In	Submitted Run(s)	Submitted Clar(s)	READY			
S1 team1	S1 team2	S1 team3	S1 team4	S1 team5	S1 team6	S1 team7	S1 team8
S1 team9	S1 team10	S1 team11	S1 team12	S1 team13	S1 team14	S1 team15	S1 team16
S1 team17	S1 team18	S1 team19	S1 team20	S1 team21	S1 team22		

At the bottom of the interface, there are buttons for Clear, Reload, All Sites (dropdown), and a checkbox for Show enabled teams.

8.2 The Runs Display

The **Runs** tab on the Run Contest tab on the Administrator screen displays a grid showing all the runs which have been submitted so far in the contest (from all teams, at all sites). The run display grid can be sorted on any column by clicking in the column header; clicking multiple times toggles the sort between ascending and descending order. The columns can also be resized by moving the column separators in the header. An example run grid is shown below.



The screenshot shows the PC^2 Administrator 1 (Site 1) [STARTED] 9.2-2298 interface. The top bar displays the time 4:55:47 and an Exit button. The main interface has a 'Run Contest' tab selected, with sub-tabs for Options, Reports, Runs, Sites, Standings, Standings HTML, and Team Status. Below these are sections for Connections, Clarifications, Finalize, and Logins. A grid of 8 runs is displayed, with columns for Site, Team, Run Id, Time, Status, Suppressed, Problem, Judge, Balloon, and Lang. The runs are all from Site 1, team5, and have a status of NEW. The problems are Sumit, Bowling for Crabs, and Problem Solved. The bottom of the interface has buttons for Give, Edit, View Judgements, Request Run, Rejudge, Filter, Auto Judge, and Extract.

Site	Team	Run Id	Time	Status	Suppressed	Problem	Judge	Balloon	Lang
Site 1	team5	1	17	NEW		Sumit			Java
Site 1	team5	2	35	NEW		Sumit			Java
Site 1	team5	3	36	NEW		Bowling for Crabs			Java
Site 1	team5	4	36	NEW		Bowling for Crabs			Java
Site 1	team5	5	37	NEW		Bowling for Crabs			Java
Site 1	team5	6	38	NEW		Problem Solved			Java
Site 1	team5	7	38	NEW		Problem Solved			Java
Site 1	team5	8	39	NEW		Problem Solved			Java

The **Runs** display provides a number of capabilities for the Contest Administrator. One function of this display is to provide the ability to select a run which has already been judged (and hence no longer appears on the Judge's grid of available runs) and "give it back to the Judges" so that it may be re-judged. (Note that this assumes an Administrative decision to re-judge a run has been made for some reason; this is not a normal contest operation.)

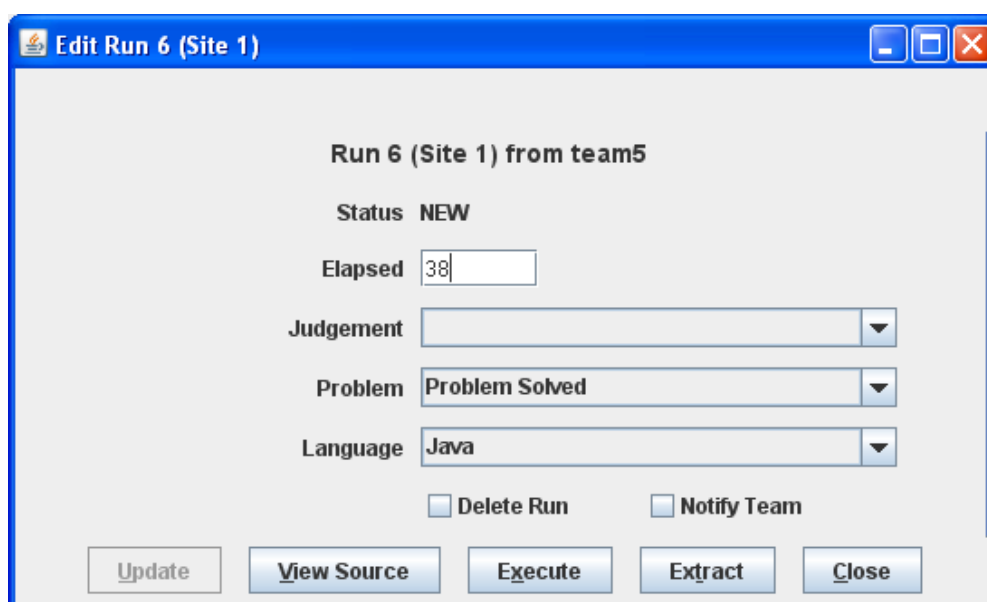
To give a run back to the Judges, click on the row containing the run to select it, then click the **Give** button. This will cause the run to appear on the Judge's screens so that it can be selected and re-judged. (Note: when a Judge selects a run which has been sent for re-judging, a warning message is displayed on the Judge's screen asking for verification that the run really is intended for re-judging.) A run does not disappear from the Administrator's grid when it is sent for re-judging; the Administrator always has a complete listing of every run submitted in the contest, from all teams at all sites.

A second purpose of the **Runs** display is to allow the Administrator to “take a run away” from a Judge. This can be used, for example, to take back a run which was given in error to the Judges for re-judging. Any time there is a run on the Judge’s display grid which should not be there (because it has already been judged and is not intended to be re-judged, for example), click on the run in the Administrator’s **Runs** display then click the **Take** button. This will remove the run from the Judge’s screens.

8.3 Editing Runs

Another function of the **Runs** display is to allow the Contest Administrator to edit various parameters associated with a specific run – for example, to mark a specific run as “deleted” or to change the effective submission time of a run. This allows the Contest Administrator to make decisions regarding unanticipated situations affecting how a run should be considered in scoring.²³

To edit a run, select the run in the **Runs** grid and then press the **Edit** button. This will bring up the following **Edit Run** dialog :



The screenshot shows a dialog box titled "Edit Run 6 (Site 1)". Inside the dialog, the text "Run 6 (Site 1) from team5" is displayed. Below this, the "Status" is set to "NEW". The "Elapsed" field contains the number "38". There are three dropdown menus: "Judgement" (empty), "Problem" (set to "Problem Solved"), and "Language" (set to "Java"). Below the dropdowns are two checkboxes: "Delete Run" and "Notify Team", both of which are unchecked. At the bottom of the dialog, there are five buttons: "Update", "View Source", "Execute", "Extract", and "Close".

The “Problem”, “Language”, and “Judgment” drop-down lists allow the Administrator to alter the specification of the corresponding attributes associated with the run. It is allowable to change multiple attributes of a run during a single edit (although this would be unusual; normally, editing a run is done for a specific purpose such as correcting a judging error).

²³ It is assumed that the Contest Administrator understands the ramifications of changing run attributes; for example, that changing the Elapsed Time affects the number of penalty points assigned to the run, changing the Problem affects how many runs a team has submitted for a given problem, etc. PC² is not smart enough to decide whether you *should* change the attributes of a run; it just gives you the *capability* to do so.

Changing the “Problem” attribute will have the effect of changing the way in which this run is considered in scoring: it will be counted as a run for the newly specified Problem (however, this will only have an actual effect on the scoring results if the Team has correctly solved the newly specified Problem; see the chapter on the Scoreboard for details). Changing the “Language” attribute will have the effect, if the run is subsequently re-executed, of changing the language definition (compiler invocation) used to compile and then execute the run. Changing the “Judgment” attribute will have the effect of causing the newly specified judgment to be the one used by the Scoreboard in determining whether the Team has correctly solved this problem.

“Elapsed Time” represents the elapsed time in minutes from the start of the contest (contest elapsed time, not counting minutes during which the contest clock was stopped) at which the run was received. The Administrator can change this value as desired. Note that Elapsed Time is considered the “team submission time” and is used to determine the calculation of penalty points assigned to the run.

Checking the “Delete Run” box will cause the run to be completely ignored in all scoring computations. Elapsed Time, Judgment value, and all other attributes of a run which is marked “Deleted” have no effect on scoring. A run marked as deleted no longer shows on the appropriate Team’s grid. (“Deleted” runs do not actually get removed from the database (nor from the **Runs** display), they are simply *marked* as such to indicate they should be ignored for scoring purposes.) If a run has been previously marked as deleted and subsequently the Mark Run as Deleted box is *unchecked*, the run is once again considered in further scoring computations, with no indication that it was previously “marked as deleted”.

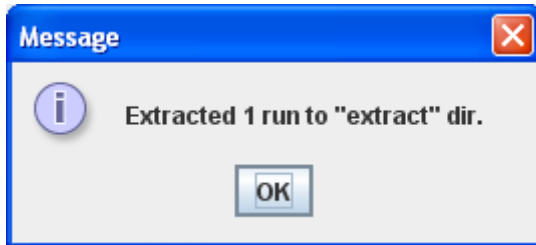
The “Notify Team” checkbox is used to indicate whether or not the Team which submitted this run should be sent a notification of the “Judgment” value applied to this run after it has been edited. Normally, editing involves correcting an internal administrative error and it is not desirable to notify the Team of any editing, so the default operation is to suppress Team notification. If it is desired that the Team *should* receive a notification of the result of editing the run (for example, a judgment was changed from NO to YES and the Contest Administrator wishes the Team to know this), uncheck the “Suppress Team Notification” checkbox.

The **Execute** button allows the Contest Administrator to execute a run just as it would be executed on a Judge’s machine. This is useful, for example, when a Team submitted a run with an incorrect language specification (which most likely caused a Judge to render a “Compilation Error” judgment for the run), and it is desired to determine what the result would have been if the language specification had been correct. The Language drop-down list can be used to change the language attribute of the run, and it can then be executed at the Admin workstation. Note, however, that *the Execute function will only work correctly on the Admin workstation if the workstation has been configured with the necessary language compilers, in the same way as on a Judge’s machine.*

Once a run has been edited (and re-executed if desired), pressing the **Update** button will store the new specification of the run’s attributes in the database and, if team notification has *not* been suppressed it will send a notice of the run status to the Team. Note that once an update has been applied, the former state of the run is lost; there is no way to restore a run to a prior state once it has been edited (other than simply re-editing the run and changing the values back – but PC² does not keep track of the old run state once the **Update** button has been pushed).

8.3.1 Extracting Run

The **Extract** button will export the run being edited. When the Extract button is clicked the following will appear:



The extract directory is under the PC2HOME directory, for site 1 run 2 the following files would be created:

```
extract/site1run2/pc2.run2.txt  
extract/site1run2/Prac.java
```

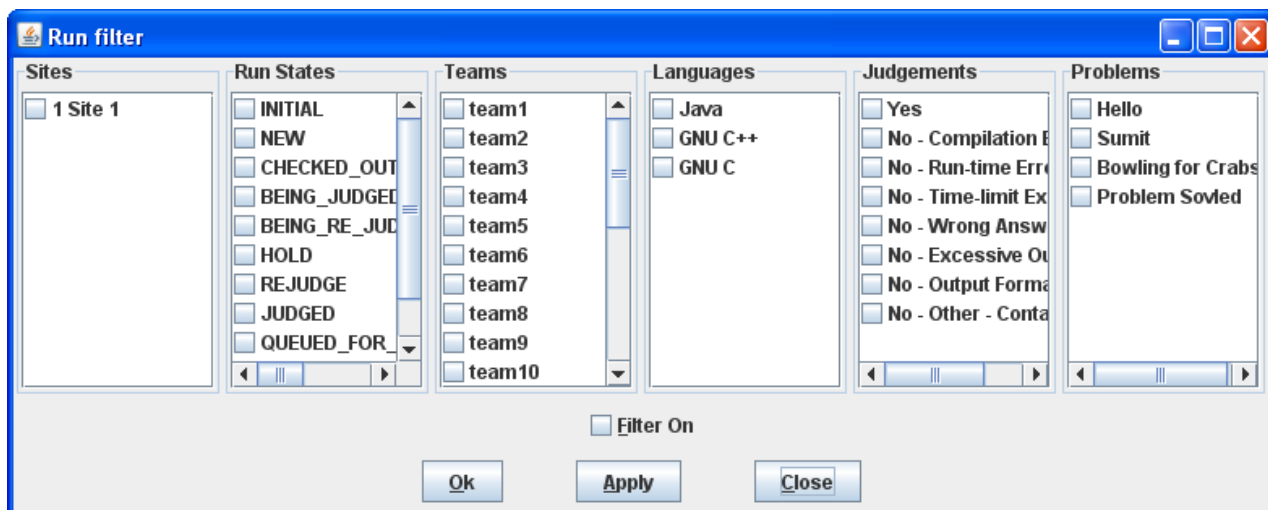
The pc2.run2.txt contains information about the run including Run #, Site #, Team Id, Problem Name, Language Name and contest elapsed time for example:

```
...  
Run   : 2  
Site  : 1  
Team  : TEAM12 @ site 1  
Prob  : Hello  
Lang  : Java  
Elaps : 4  
...
```

8.4 Filtering Runs

It is frequently desirable to view a *subset* of the complete set of runs which are currently in the database. For example, the Contest Administrator may be interested in looking at all the runs for a given Problem, or all the runs from a given Team, or all the runs submitted during a specific window of time, or some combination of these.

The Administrator **Runs** display has associated with it a *filter* which can be used to apply a set of filtering criteria to the runs which are displayed.²⁴ Pushing the **Filter** button on the **Runs** display screen will produce the following dialog, which is used to set the filtering criteria:



Selecting a set of items in the filter dialog indicates that those items *should* be displayed on the **Runs** grid. For example, to specify that the **Runs** grid should display all (and only) runs from Team 1 at site “CSUS” for the problem named “Penguins”, you would select “Penguins” in the Problems column, “Team1” in the Teams column, “CSUS” in the Sites column, and “All” in the Language, OS, and Time columns, and then press the **Update** button. The **Runs** grid would then apply the specified filter criteria to all runs, displaying only those that match the selected criteria. As another example, to display all (and only) runs which were submitted during the first 20 minutes of the contest, you would enter “0” in the “From” time field, “20” in the “To” time field, and select “All” in the Problems, Teams, Sites, Language, and OS fields, and then press the **Update** button.

The filter is designed with the expectation that at least one entry will be selected in each column. The “Clear All” button in each column is provided as a convenience to allow “de-selection” of all entries in anticipation of then selecting one or more entries in the column. However, clearing all criteria in a column does not make sense as a final filtering operation: if “Clear All” were allowed as the final choice in a column (that is, if no entry in the column was selected following a “Clear All” for the column), this would imply that *no* runs should be displayed (since no runs would match the filtering criteria of “nothing selected”). For this reason, the filter ignores a “Clear All” condition if it is the last operation performed on a column and

²⁴ The Judge client provides a similar filtering operation; the discussion here on filters applies equally to the Judge.

instead effectively selects all the criteria in that column. In other words, pushing “Clear All” as the last operation on a column is effectively the same thing as pushing “Select All” on that column.

The same logic applies to the “Clear All” button near the bottom of the screen, which is used to initially clear all entries in *all* columns, in anticipation of then selecting at least one entry in each column. If this global “Clear All” is the last operation selected prior to pressing the **Update** button, the filter interprets this as a request to *disable the filtering operation entirely*; that is, to display all runs on the **Runs** grid. Thus, the global “Clear All” button is effectively a “Filter Off” (no filtering) button.

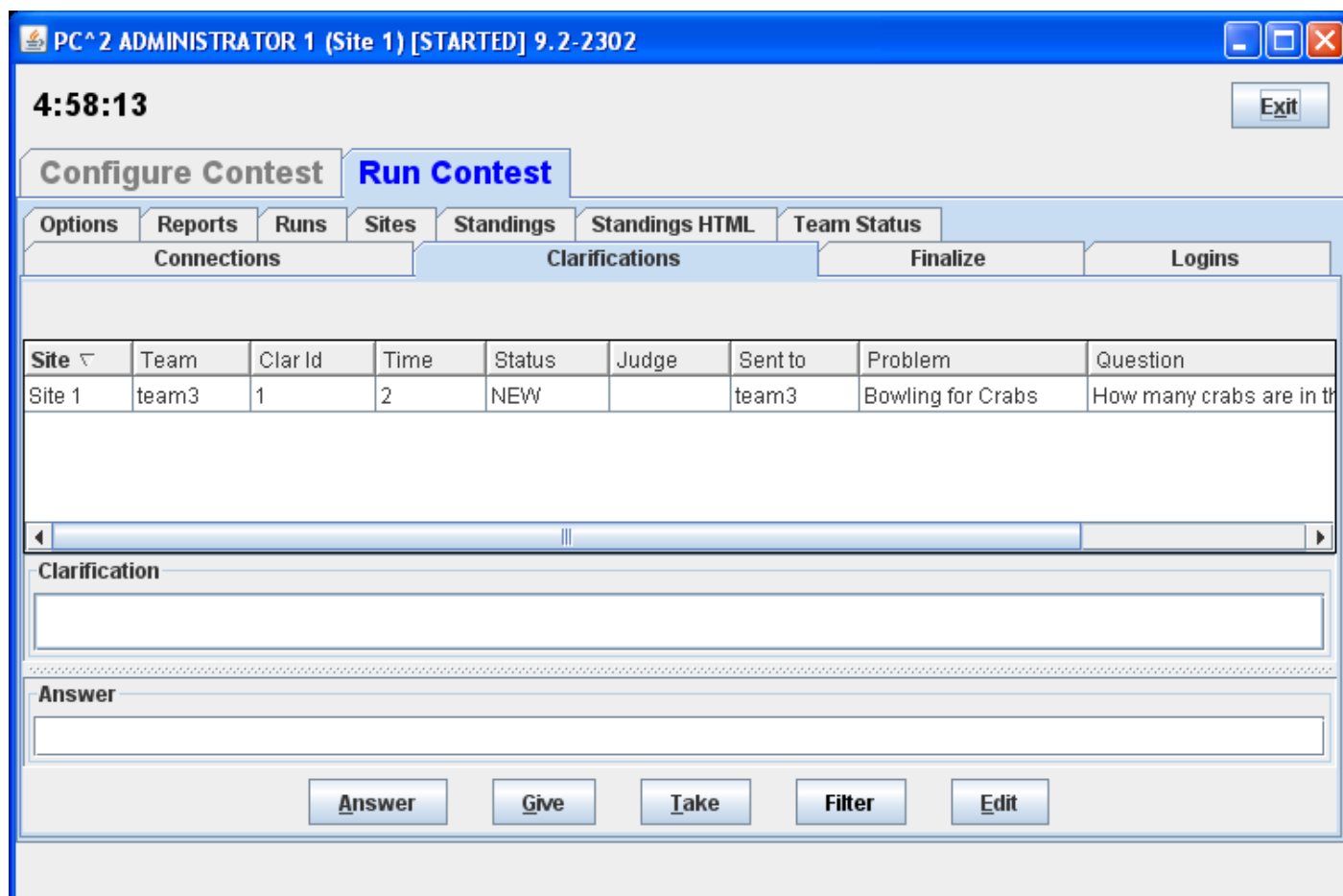
In order to remind the user when filtering of runs is taking place, any time a filtering operation has been selected the “**Filter**” button on the Admin screen will change color to blue and will indicate “on”, like so: **Filter (ON)**. Disabling filtering (using the “Clear All” button as described above) will return the **Filter** button to its normal state.

The “Ok ” field is used to specify the command line which is used to compile source code and produce an “executable program file” in the language.

The **Ok** button will save the filter settings and if the filter is ON will update the Runs list with the filter settings.

The **Apply** button will immediately reload the Runs list with the filter settings (if Filter On is checked). This can be used to quickly find runs based on the filter instead of having to use 3 steps to see the filtered results. Three steps would be: 1) (Edit) Filter, 2) change filter 3) **Ok** button

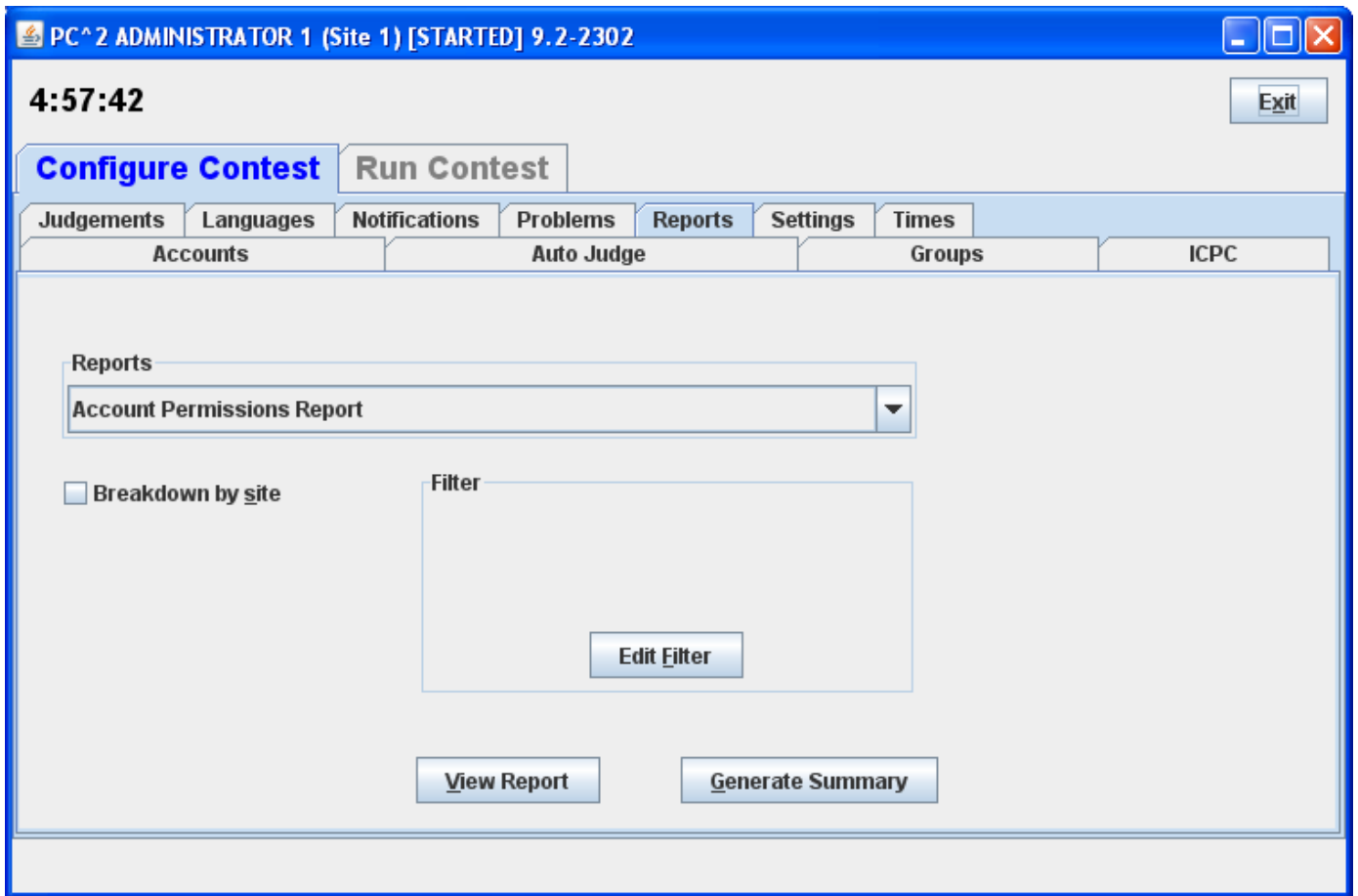
8.5 Clarifications



The **Clars** tab on the main Administrator screen displays a grid showing all the Clarification Requests which have been submitted so far in the contest (from all teams, at all sites), in a format similar to the **Runs** grid. Like the **Runs** grid, the Clarification Request grid can be sorted and resized by manipulating the columns headers. The **Give** and **Take** functions work like the **Give** and **Take** on the Runs pane.

8.6 Reports

The **Reports** tab on the main Administrator screen provides a variety of options for generating statistical reports about a contest, both during and after the contest. The **Reports** screen looks like the following:



The “**Report Option**” drop-down list allows choosing one of a number of different report formats. The list of available reports is summarized in the table shown below. Pressing the “**View**” button will pop up a display showing the content of the selected report, and will also write the selected report in text form to a file (located by default in the \$PC2HOME directory) using the file name given in the “**Report File Name**” text box.

Generate Summary will generate these 17 reports: Balloons Summary, Clarifications, Contest Analysis, Contest Settings, Contest XML, Evaluations, Fastest Solutions Per Problem, Fastest Solutions Summary, Languages, Problems, Runs, Runs (Version 8 content and format), Runs grouped by team, Solutions By Problem, Standings Web Pages, Standings XML, Submissions by Language.

When all reports are written a dialog will appear that shows the directory where the reports were saved. Reports will be saved in the `profiles\P<profileid>\reports` directory.

Automatic Generation of Reports at End of Contest

When a server is shutdown and there are less than 30 minutes remaining in the contest, 17 reports will be automatically generated in the reports directory.

Report	Description
Account Permissions Report	For each client list their Permissions/Abilities
Accounts	List summary of accounts per site, and individual accounts sites, logins, passwords.
All Reports	List contents of all reports
Balloons Delivery	List of all balloon deliveries by team, by problem and time of delivery
Balloons Summary	List summary of which teams should have which color balloons.
Clarifications	List all clarifications
Client Settings	Various settings like Notification settings
Contest	Contest settings in XML (work in progress)
Contest Analysis	Summary of submissions, unjudged runs, various checks on runs.
Evaluations	One line per judgment output
Extract Replay Runs	Files extracted used with Replay feature
Fastest Solved by Problem	List all run solving problems by fastest, by problem, and fastest solution showing rank, elapsed, team name
Groups	List of Groups (Regions)
Internal Dump	An internal dump of a bunch of config settings
Judgement Notifications	List (balloon) notifications grouped by problem.
Judgements	List of judgments
Languages	List Languages
Logins	List who is logged in
Notification Settings	List of Notification Settings
Problems	List problems
Run 5 field	List of runs: run #, team #, problem letter, elapsed time, judgment
Run Notifications Sent	

Report	Description
Runs	List of runs, with run#, run state, team #, team name, whether judgment sent to team and details on each judgment
Runs (Version 8 content and format ²⁵)	List of runs with detail (see below)
Runs grouped by team	List of runs, grouped by team, then by problem, helpful in calculating scoring.
Solutions By Problem	For each problem show number of run with No, Yes, and percentage correct
Standings XML	Standings in XML format
Submissions by Language	A summary of how many teams used which languages.

²⁵ Example single line of output: run 59|site 1|proxy |team 11|team11:Crimson Pride (WSU)|prob D - Obstacle Course-6938491730217106684:D - Obstacle Course|lang GNU C++-1782097724418977383:GNU C++|tocj |os Linux|sel false|tocj false|jc true|182|rid Run-2641274401129060175|mmfr true|del? false|jt 0|jby judge5|jci Yes|

9 The PC² Scoreboard

9.1 Overview

PC² contains a separate “Scoreboard” module which keeps track of the current standings in a contest. The scoreboard provides several functions: it automatically generates HTML pages describing the current state of the contest in a variety of formats; it generates email and/or printed “balloon notifications” (when that option has been selected by the Contest Administrator; see the section on “Options”, above); and it provides the capability to generate an “export” file containing contest standings data (in the form required for importing to the ICPC Registration system).

The HTML files generated by the scoreboard are placed in a directory named “html” (lower case) directly beneath the current working directory. Balloon notifications are sent to destinations as configured by the Contest Administrator.

The following sections describe the details of the generated HTML files and the export data file. They also describe the overall scoreboard operation, including the scoring algorithm used to compute contest standings.

9.2 Starting the Scoreboard

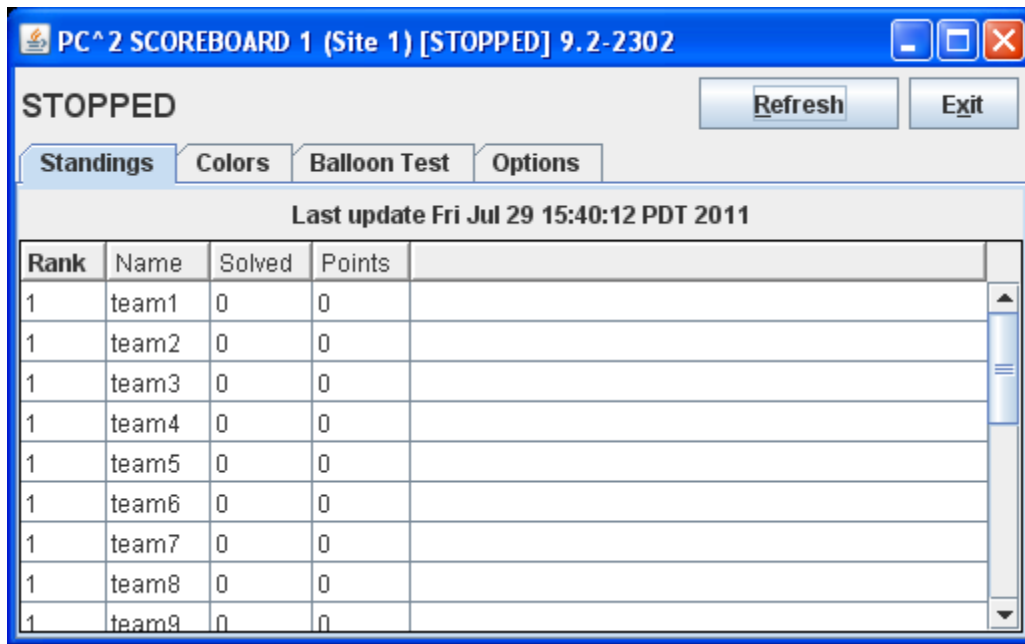
To start a scoreboard, go to a command prompt and type the command “**pc2board**”. This will start a PC² client expecting a scoreboard login.

Once the Client login window appears, enter the scoreboard account name and password of the Balloon Client as defined when PC² accounts were created²⁶. This will bring up the PC² Scoreboard display window (next page), indicating that the scoreboard program is running.

The scoreboard automatically generates a complete set of HTML files to the **html/** directory as soon as it is started. Thereafter it generates updated HTML files periodically according to an algorithm described below (See Scoreboard Updates). Each time a new set of HTML files is generated the scoreboard display window is updated to show the most recent update time.

Under normal circumstances it is only necessary to have a *single* PC² scoreboard running, even in a multi-site contest. The scoreboard automatically receives update information from every site server, and generates HTML files describing the overall contest status (including all sites). These HTML files can be copied to a publicly-accessible location for access by a browser (see below), so participants at any location can see the current standings. In addition, a single scoreboard can generate balloon notifications for all sites. Thus there is rarely a need for running more than one PC² scoreboard in a contest, and this is the recommended mode of operation.

²⁶ Recall that by default, account passwords are the same as the account name. However, if the scoreboard account password(s) are not changed by the Contest Administrator, it would mean that any team could start a scoreboard running on their own machine, allowing them to look at the contest standings even during times when the Contest Administrator has decided to hide that information (such as near the end of the contest, which is the policy in some contests). We strongly recommend *changing* the scoreboard account passwords.



9.3 Scoreboard Updates

Once the scoreboard is running, it waits passively until some contest event occurs which could alter the data it should display (this could be a submitted run, a judgment decision by the Judges, a complete list of events is below). When any such event occurs, the scoreboard obtains from the server an update of the contest state, computes the new standings based on this information, and regenerates the HTML display files.

Events that will cause a scoreboard update: run judgment, run submission, account name changes, group name changes, scoreboard settings changes and lastly if a user clicks the **Refresh** button on the board display. The date/time of the last scoreboard update is present on the scoreboard main window.

Scoreboard Refresh Events

All of the following events/changes will cause a scoreboard files Refresh.

1. the scoreboard module is started
2. the Refresh button is clicked.
3. any run is judged
4. any run judgment is changed (via Edit Run)
5. configuration definitions/settings are changed (accounts, languages, problems, etc.)

9.4 Scoreboard HTML Files

Each HTML file generated by the PC² scoreboard is a complete stand-alone HTML document (i.e. is bracketed by `<html> ... </html>` tags). Each document `<head>` includes a `<title>` tag, into which PC² places the Contest Title as specified by the Contest Administrator (see **Options**, above). Each document `<body>` contains an imbedded `<table>` holding contest status information. Each HTML file is also in an XML formatted file.

The `<table>` in each different HTML file contains a different set of contest information, such as team rankings, run submission statistics, etc. (see below). The information outside the `<table>` can be edited/replaced as desired, for example by adding additional header information, frames, or any other HTML constructs. However, it is important to keep in mind that the set of HTML files is *completely regenerated* on every scoreboard update; changes made manually to an HTML output file will only persist until the next scoreboard update.

The following HTML files are always generated by the scoreboard:

File Name	Table Contents
full.html	Columns showing rank, team display name, number of problems solved, and penalty points, with rows ordered by rank. This is the standard “contest standings” display.
fullnums.html	Same as full.html except that the Team Display Name is preceded by the Team Number followed by a dash.
final.html	Standings as they are displayed at the conclusion of the ACM ICPC World Finals – Penalty Points are considered and displayed only for the top twelve teams; the remaining teams who have solved at least the median number of problems are grouped alphabetically by number solved; the rest of the teams are listed as “Honorable Mention”.
sumtime.html	A grid showing, for each team and each problem, the number of runs submitted by the team for the problem, and, if the team has solved the problem, giving the contest elapsed time of the team’s solution.
sumatt.html	A table similar to the sumtime table but instead of giving the time of solution for solved problems simply indicates “Y” or “N” according to whether the team has solved the problem.
summary.html	A table combining the full and sumtime displays described above – that is, a showing rank, team display name, number of problems solved, penalty points, and number of runs submitted and solution time for each problem, with rows ordered by rank.

The most common way to take advantage of the HTML files generated by the PC² scoreboard is to run a separate external process (e.g. a batch script²⁷) which repeatedly copies the

²⁷ A sample batch script is included with the samples: `samps/web/distribute_score` .

current HTML files to some external location (web site), reformats them if desired, and makes them available to teams and spectators using a browser.

There are several advantages to this method of operation. First, the details of the appearance of the scoreboard can be customized by the Contest Administrator external to PC². Thus the Contest Administrator can choose to take advantage of the full set of scoreboard screens, or can choose to omit some or all of them. In addition, it is not necessary for teams, judges, spectators, etc. to run separate PC² scoreboards, since most users will already have access to a browser. The Contest Administrator can arrange that the external scoreboard script builds the desired scoreboard display and puts the resulting HTML in a standard public location accessible to all user's browsers.

9.5 Scoring Groups

In addition to the above files, the scoreboard can generate separate HTML files showing rankings based on the concept of "groups" or "regions" with which a team is associated. For example, in the ICPC World Finals, teams compete not only for placement in the overall world-wide standings, but also among teams from their own region of the world for the regional championship.²⁸ Other examples include situations where it is desirable to break contest teams up into separate groups based on level of experience (e.g. "lower division" and "upper division" students), and in multi-site contests where it is desirable to be able to display rankings that show only those teams participating at a given site.

Every PC² account has associated with it a "Group ID" identifying the "region" or "group" to which the account belongs. By default all accounts have a Group ID of zero (which is considered a valid Group ID); hence all teams are by default in the same "region" (group). Changing the Group ID for a particular team associates that team with other all teams having the same Group ID (see the section on Editing Accounts, earlier in this manual).

By default, the scoreboard ignores Group IDs. However, it is possible to make the scoreboard pay attention to Group IDs (that is, pay attention to team groupings), and when this is done it will cause the scoreboard to automatically generate additional HTML files, as follows:

File Name	Table Contents
regions.html	A single table showing the top team in each region (team grouping). ²⁹

²⁸ Currently the ICPC defines six world "regions" (called "Super Regions" for historical reasons), and awards the Super Regional Championship to the top team in each Super Region as part of the Word Finals competition each year. The current ICPC Super Regions are: Africa and the Middle East, Asia, Europe, Latin America [comprising Central and South America], North America, and the South Pacific.

²⁹ Note: if multiple teams are tied for the top position in a region after taking into account number of problems solved, penalty points, and time of last correct submission, the region entry in the **regions.html** file will simply display <TIE> without displaying a team. If this happens, check the **regionX.html** file to determine the standings.

File Name	Table Contents
groupX.html	This represents several different HTML files, one for each group “X” (group1.html , group2.html ,....). Each groupX.html file contains the same contents as the file full.html described above, except that the data is restricted to teams which are defined as belonging to “group X”. ³⁰

There are two ways to cause the scoreboard to pay attention to regional groupings of teams (and hence generate the additional region HTML files). One is to use the “import ICPC data” function on the main Administrator screen. The imported ICPC data contains information identifying for PC² both the number of regions into which teams have been partitioned, and the “region ID” and “region name” associated with each team. This information both enables the automatic generation of the regional HTML files, and tells the system how determine the region with which a given team is associated.

The Contest Administrator must also take care to assign a Group ID to each account according to the region to which the account belongs. All accounts in the contest (even accounts across multiple sites) which belong to the same region or group should have the same Group ID number, since this is how the scoreboard determines the association between an account and its region. (This assignment is handled automatically when the Import ICPC data operation is used.)

For example, if it was desired to separate all contestants into “Lower Division” and “Upper Division” groups, all accounts used by teams in the “Lower Division” might be given Region ID values of “1” while all accounts used by teams in the “Upper Division” might be given Region ID values of “2”. Note that in a multi-site contest, all Lower Division teams would need to have the same Region ID (2), regardless of the site at which they were located.

9.6 Scoring Algorithm

The algorithm used in PC² to compute Rank and “Score” (Penalty Points) is the one used in the ACM ICPC World Finals, which is as follows:

- 1) Teams are ranked according to the number of problems solved; a team solving more problems is always ranked higher than a team solving fewer problems.
- 2) Within a group of teams solving the same number of problems, teams are ranked by increasing “Penalty Points” (that is, the team with the lowest number of Penalty Points is ranked highest within the group). Teams only accrue Penalty Points for problems which the team has **solved**; unsolved problems do not affect the scoring in any way. Teams accrue Penalty Points for solved problems in two ways:

³⁰ In the current version of PC² this description is taken literally: the data in each **regionX.html** file is the same as that which appears in the **full.html** file. In particular this includes ranking numbers; thus, **regionX.html** files display a team’s ranking in the overall contest, not just in the region as might be expected. However, teams will be ordered in descending rank order so it is easy to discern relative rankings.

- One point for each minute elapsed from the start of the contest until the problem was solved (the time of SUBMISSION is counted as the “time solved”; it does not matter how long it took the Judges to judge it).
 - A specific number of penalty points for each INCORRECT submission submitted to the Judges *prior to a correct solution* for the problem (runs submitted after a correct solution are not counted in the scoring).³¹
- 3) If two or more teams have the same number of solved problems and exactly the same number of Penalty Points, ties are broken in favor of the team with the earliest time of the last correct submission (that being the time when the team “finished” the contest).

Examples of the PC² scoring algorithm can be found in the PC2 Wiki article Scoring Algorithm at http://pc2.ecs.csus.edu/wiki/Scoring_Algorithm.

9.7 Adding new HTML Files

New HTML (or other format) files can be created to augment the existing files. PC² uses two inputs to create output scoreboard HTML. The first input is an XML Style Sheet (xsl) user supplied file in xsl directory, the second is the system supplied standings XML file.

Steps to creating and generating HTML Files

1. Create a new XSL file
2. Copy the new XSL file into the scoreboard `$PC2HOME\data\xsl` directory
3. Click Refresh on the scoreboard module.

On Refresh the scoreboard module will read each file with `.xsl` extension in the `data\xsl` directory and create `.html` file in the `html` directory.

For example the `full.xsl` file will create the `full.html` file, the file names and path are:

```
data\xsl\full.xsl
html\full.html
```

There are 5 xsl files that create HTML files in the html directory:

```
data\xsl\full.xsl
data\xsl\fullnums.xsl
data\xsl\sumatt.xsl
data\xsl\summary.xsl
data\xsl\sumtime.xsl
```

There are sample XSL files in the `samps\web\xsl` directory that create region HTML files and World Finals HTML, see the file `samps\web\xsl\README` for more details or send the pc2 group email if you have questions.

³¹ In the ICPC World Finals the number of penalty points for each incorrect submission prior to solving a problem is always 20; this is also the default value in PC² (although PC² allows the Contest Administrator to change this value using a Configuration Option).

9.8 Export Data File

The scoreboard provides the ability to generate a text file containing data describing the current contest standings. Since this capability was provided for purposes of exporting contest results back to the ICPC Registration system, the button on the scoreboard display which is used to invoke it is labeled **Refresh**. However, the use of the export operation is not restricted to interfacing with the ICPC Registration system; it simply generates a file containing text records made up of comma-separated fields.

The export data file contains, for each team, a record giving the number of problems solved, the total number of penalty points accrued on solved problems, and the time of last submission of a correct solution (used in the ICPC World Finals as a tiebreaker). See the Appendix on Import/Export Interfaces for further details on the format of the export data file.

The export data file, **pc2export.dat**, is created in the (current working) directory where the scoreboard was started.

In Version 8 the export data file was generated by pressing the **Export ICPC** button the Scoreboard.

In version 9 the export data file is generated by pressing the **Refresh** button on the Scoreboard. The export data file is regenerated each time the scoreboard HTML is updated.

10 Shutting Down

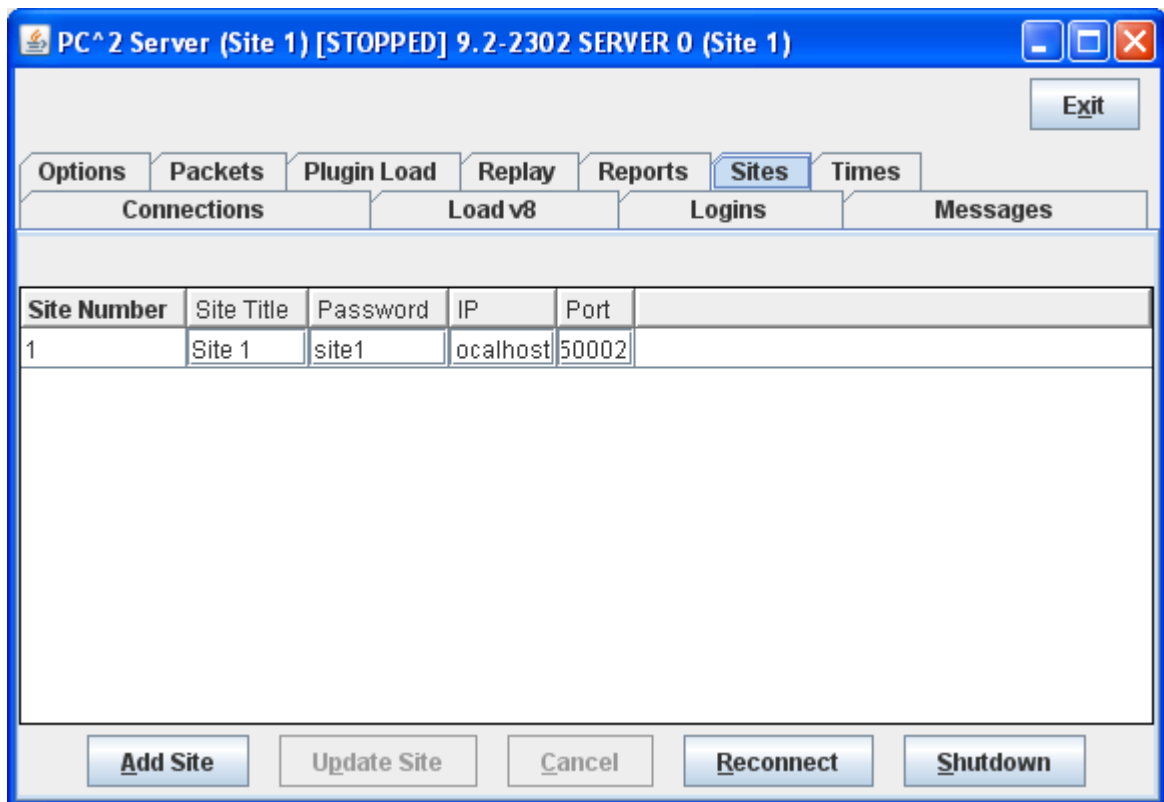
10.1 Shutting down a server

Each site server can be shutdown by using the Exit button on the Server GUI. When a server is shutdown *all logged in clients* will also be shutdown.

For non-GUI servers use the Shutdown button on the Site tab on the Administrator to shutdown the server.

10.2 Shutting down all servers

The Shutdown button on the Site Tab can also be used to shutdown a remote server or all servers.



Appendix A – pc2v9.ini Attributes

As described in the chapter on PC² Initialization Files, the `pc2v9.ini` file consists of `[server]` and `[client]` sections, with each section containing one or more "attribute assignment" statements of the form `attributeName=value`. Lines in the file which begin with a “#” or “;” character are ignored, as are blank lines. Attribute names (left side of the equal-sign) are not case sensitive; however, string data on the right side of the equal-sign *is* case sensitive.

The following list gives the attributes which can be defined in each section of the `pc2v9.ini` file, along with a description of their function. Some attributes may appear in more than one section.

[server] section attributes:

`port=<portNumber>`

Tells the server the port number on which it should expect to be contacted by clients, and by other PC² servers in a multi-site contest. This attribute may be omitted from the `pc2v9.ini` file, in which case it defaults to 50002. Note that if you choose to assign a specific port number, then all clients and other servers contacting this server must also be told to use this same port number (this is specified with the “`server=`” attribute in the case of clients, and with the “`remoteServer=`” attribute in the case of other servers). Note also that if you choose to assign a specific port number, the port number should be greater than 49151 according to the conventions established by the IANA (Internet Assigned Numbers Authority)³².

`remoteServer=<IPAddress>:<portNumber>`

Tells the server the IP address and port number of a remote PC² server at another site which it should contact in order to join a multi-site contest. The `<portNumber>` must be specified and must match the port number being used by the server at the remote site. The appearance of this attribute makes this server a “secondary” server; if this attribute is not defined in the `[server]` section then this server is a “primary” server and waits passively to be contacted by other site servers.

³² <http://www.iana.org/assignments/port-numbers>

[client] section attributes:

server=<IPAddress>:<portNumber>

Specifies the IP address and port number at which the client module should contact the PC² server. Every client module **MUST** have a “**server=<IPAddress>:<portNumber>**” entry in the **[client]** section of its **pc2v9.ini** file. The IP address and port number must correspond to the address of the machine running the PC² server and the port number at which the server on that machine is expecting to be contacted.

plaf=<type>

Specifies the “Programmable Look-And-Feel (PLAF)” which should be used in displaying the client GUI. Allowable values for **<type>** are “**java**”, which causes the GUI to use the standard Java GUI appearance (which means that client GUIs will look the same regardless of the underlying platform), and “**native**”, which causes the GUI to use the underlying platform’s “native look-and-feel” – so for example on a Windows machine the GUI will look “Windows-like” while on a Mac the same GUI will look “Mac-like”.

Appendix B – Networking Constraints

As mentioned in the beginning of this manual, PC² modules must be able to communicate with each other via TCP: clients must be able to communicate with their servers, and servers in a multi-site contest must be able to communicate with other servers. If client machines reside on the same segment as their server, and if all servers have publicly routable IP addresses which can be reached by other servers, then communication should work with no problems.

However, given the wide variety of network configurations which can exist – firewalls, NAT, and VPNs, just to name a few – there may be some constraints in a given network setup which will cause problems in setting up a contest using PC². In order to understand how to avoid (or circumvent) these problems, it is useful to have some understanding of how PC² networking is implemented.

PC² is written in Java and uses TCP sockets, each server module requires an IP and port which is not blocked by firewall. Server modules listen for incoming connections using port 50002.³³ Client machines initiate connections to servers and servers initiate connections to other servers.

Another constraint on networking has to do with NAT (Network Address Translation). For PC² to work using NAT you must configure port forwarding on your firewall, and configure the site table with the public address/port for remote connections.

In a multi-site contest each server contacts every other server. There must not be a firewall that blocks the sending/receiving of data over the required listening ports.

³³ The listening port is configurable.

Appendix C – PC² Server Command Line Arguments

The command to start a server is:

```
pc2server
```

The server accepts a number of command line options when it is started. One option is **-h** (or **--help**), which produces the following “usage” output:

```
$ pc2server -help
Usage: Starter [--help] [--server] [--first] [--login <login>]
      [--password <pass>] [--site ##] [--skipini] [--ini filename]
```

As seen from the “usage” output, the server also accepts the following command line options:

- first : indicates that this server is a primary server and should not attempt to contact any other servers (ignores any **remoteServer=** attribute in **pc2v9.ini**)
- join : indicates that this server is a secondary server and should attempt to contact a remote server to join a contest. Uses the **remoteServer=** attribute in **pc2v9.ini**, if there is one, to determine what remote server to contact; if no attribute is defined, prompts at the console for IP:port connection information
- server: indicates that this Starter is to run as a server, otherwise starts as a client.
- contestpassword: on the first server only, specifies the contest password.
- F : specify a text file with command line options, an alternate to specifying sensitive information on the command line. See section Using the -F option for more details.
- login : specify the PC² login account name
- password : specify the PC² password
- site : (Not used)
- skipini : ignore the pc2v9.ini file
- ini url_to_ini_file : specify an override ini filename

Using the -F option

The -F command line option will load command line options from an input text file. This option is a security feature. Under most Unix systems the complete command line is listed when using a ps or similar command revealing login id's and passwords. Using the -F option, login id's and passwords can be stored in a text file. Note that the command line options are not limited to login and password options, any command line option can be stored in the specified text file.

If this command line was used:

```
pc2server --nogui --contestpassword cpass --login sitel --password sitelpass
```

One could alternatively use the -F option:

```
pc2server -F secure.txt
```

where the secure.txt file contains

```
#  
# Command line for non GUI server  
#  
--nogui  
--contestpassword cpass  
--login sitel  
--password sitelpass
```

Blank lines and lines starting with # are ignored in the file (secure.txt).

The order that command line values are applied (highest precedent first) are:

1. specified -F option properties
2. specified on the command line
3. pc2v9.ini

Appendix D – ICPC Import/Export Interfaces

D.1 Importing ICPC Registration Data

As mentioned earlier in this manual, PC² was designed for supporting the ACM International Collegiate Programming Contest, including its local and Regional contests worldwide. The ICPC maintains an online Contest Registration system which is used by Regional Contest Directors (RCDs) around the world to manage participation in the various ICPC Regional Contests. PC² provides interfaces to import contest registration data from the ICPC Registration system, and also to export contest results back to the ICPC web site.

To import ICPC Registration data to PC², the RCD must first log into the ICPC Registration system and download the “PC² Initialization” zip file which is automatically created and updated as registration data changes occur.³⁴ Once the PC² Initialization data file is downloaded, it should be “unzipped” at any convenient location.

Unzipping the PC² Initialization data file will produce three separate files: “PC2_Contest.tab”, containing details about the organization of the contest (such as the formal name of the contest); “PC2_Site.tab”, containing data identifying the sites in the contest; and “PC2_Team.tab”, containing data about the teams that are registered in the contest.

The PC2_Team.tab file consists of text-based tab-delimited records, one record for each team registered in the contest. The tab-delimited field contents of each record are as follows:

- 1) An integer giving the ICPC Team ID (note that this is *not* the same thing as the PC² Team ID; see below).
- 2) An integer giving the ICPC Region ID (the “region” or “group” to which the team belongs).
- 3) A single character indicating the Team’s “registration status” for the contest – typically either ‘P’ (pending), ‘A’ (accepted), or ‘C’ (cancelled). PC² ignores records containing ‘C’ in this field.
- 4) A string giving the Team Name; for example, “The Top Coders”.
- 5) A string giving the full name of the Team’s school; for example, “California State University, Sacramento”.
- 6) A string giving the Team’s school name in “short form”; for example, “CSUS”.
- 7) A string giving the Team’s school’s URL; for example, <http://www.csus.edu>.
- 8) A string giving the Team’s school’s country code (three letters); for example, “USA”.
- 9) A single character ‘Y’ or ‘N’ indicating whether the Team’s school has a graduate program. PC² ignores this field (but it must be present).

³⁴ The PC² initialization data is contained in a file whose name on the ICPC web site is typically something like “CI532.zip” – “Contest Information for contest number 532”. However, both the naming convention for the file, and the exact location of the file on the web site, are outside the scope of (i.e., not controlled by) PC².

PC² can “import” this team data and use it to define things such as the team’s Display Name on the scoreboard. PC² will associate consecutive records in the `PC2_Team.tab` file with consecutive PC² accounts – the first record in the file will automatically be associated with “team1” at the site specified in the record, etc. (Note that PC² does *not* use the ICPC Team ID field for purposes of identifying a team.)

If the contest administrator wishes to associate registered teams with PC² accounts in a different order, the `PC2_Team.tab` file can be edited (rearranged) to present the records in the desired order – the team which should be “team1” in PC² should appear first in the file, etc. Alternatively, if the `PC2_Team.tab` file is copied to a new file named “`_PC2_Team.tab`”, the new file can be edited by adding a new leftmost column containing a PC² team number. PC² checks for the existence of this file and, if it is present, uses that file not only to load the ICPC data but to determine, based on the value in the first field in each record, to which PC² account each record in the data applies.

Once the PC² Initialization data has been unzipped and the `PC2_Team.tab` file edited if desired, the next step is to press the “Import Accounts” button on the PC² Administrator ICPC tab under the Configure tab. This will cause PC² to display a “file selection” dialog; this dialog is used to identify for PC² the set of .tab files which should be imported. To make this identification, simply navigate to the directory containing the unzipped ICPC PC² Initialization data files, and select any one of the “.tab” files in that directory. This will cause PC² to load all three of the initialization files. Note: all initialization “.tab” files must all reside in the *same directory*.

If this is the first time an ICPC Import has been done, PC² will show a dialog displaying the names of the contest sites as defined in the ICPC data and asking the administrator to provide a specification of the association of those site names with the site names which PC² already knows about when they were entered on the Sites tab. Once this is complete, PC² saves the region-association information³⁵ and then finishes the association of ICPC Registration data with PC² accounts; the results can be viewed on the **Accounts** tab on the Administrator main screen, just as if the data had been entered manually. (Note: the account display on the Administrator screen is not dynamically updated upon completion of an ICPC import; click to another view and then back to the “Manage Accounts” view in order to see the imported data.)

Note that each record in the ICPC PC² Initialization data contains *multiple* names associated with each team – the team name, the full name of the school, and a short version of the school name. PC² can be configured to use any one of these names, or a combination of them, as the name to be displayed on the scoreboard. See the appendix on `pc2v9.ini` file entries, under the **[admin]** section, for details.

PC² expects quotation marks in the team data to be “quoted”. That is, if any field in the data contains a quotation mark (”), then (1) the entire field must be surrounded by an additional

³⁵ The region-site association data is saved in a file named “`_pc2_site.tab`”. If this file exists when an Import ICPC Data operation is performed, PC² does not prompt for the region-site association information, but uses the contents of the “`_pc2_site.tab`” file instead. To force it to reread new imported site information (for example on a subsequent Import), delete the file “`_pc2_site.tab`”.

matching set of quotation marks, and also (2) each quotation mark which is part of the data must be doubled. Thus for example a team name like **The "TOPS" Team** should appear in the import data file as **"The ""TOPS"" Team"**. If the PC² Initialization file exported from the ICPC web site contains data with quotes, it may be necessary edit the data by hand (or load it into a program such as Microsoft's Excel and then save it) to insure that quotation marks are properly formed. If this is not done prior to importing the data into PC², you may see "format error" messages in the log file, and the data containing the quotes will not be displayed properly.

D.2 Exporting Contest Data

Pressing the "Refresh" button on the scoreboard display causes the scoreboard to generate a text file containing the current contest standings in a form required by the ICPC World Finals results display algorithm.³⁶ This file is made up of a series of text records, one record for each team in the contest. Each record contains a set of comma-separated fields. The fields in each record are:

- 1) ICPC Team ID (note: *not* the PC² team number).
- 2) Rank in contest (this field is blank if the team falls in the "Honorable Mention" category as defined by the scoring display algorithm for the ICPC World Finals; see the description of the file **final.html** in the chapter on the PC² Scoreboard for details).
- 3) A non-negative integer giving the number of problems the team has solved.
- 4) A real number giving the total number of penalty points accrued by the team.
- 5) A real number giving the time of the last submission by the team, taking into account only the problems which the team has solved (used as the ICPC World Finals tiebreaker determination).

The records in the file are sorted by team rank based on problems solved and penalty points. In particular, teams that fall into the "Honorable Mention" category as defined by the ICPC World Finals results display algorithm will still appear in rank order in the file, as defined by number of problems solved, penalty points accrued, and tie-breaking time of last submission.

The exported data file contains all the information necessary to update the ICPC web site registration system with the results of a contest. It is primarily intended to provide an automated mechanism for Regional Contest Directors to post the results of Regional Contests.³⁷ The export data file can also be imported into any program that wishes to make use of the standings data.

Note that if no "ICPC Import" operation was performed prior to invoking the "Export ICPC" operation, then PC² will have no record of the ICPC Team ID associated with each team. In this case the "Team ID" value in field #1 in the exported file will be zero. This problem can be circumvented in contests other than Regional Contests (that is, contests where there is no ICPC data to import) by creating a local version of the "PC2_Team.tab" file, entering the appropriate

³⁶ Actually, the export data file is automatically generated (updated) each time the contest standings change; pressing the "Refresh" button simply displays a message showing its location.

³⁷ However, to our knowledge as of this writing (July 2011) the upload functionality on the ICPC web site has not yet been implemented. This is a separate operation outside the scope of PC²; RCDs should check with the ICPC Contest Manager as to the availability of this function.

PC² team account number in lieu of the ICPC Team ID. This will cause the “**Export ICPC**” operation to generate a file containing all the data necessary to compute complete contest standings utilizing PC² account numbers.

Appendix E – Validators

E.1 Overview

PC² allows the Contest Administrator to configure each problem so that it has associated with it a *validator* program whose purpose is to help automate the judging process. A “validator” is a program which is given, as input, the output of the execution of a run (that is, the output of a program submitted by a team).

Validators can operate in one of two ways: passively or actively. A passive validator is a program which accepts the team program’s output and displays it in some useful form for examination by the human judge. An active validator is a program which not only accepts the team program’s output but contains logic designed to make a determination, according to some set of rules, regarding the correctness of the output. An active validator can also return the result of its determination to PC².

An example of a passive validator would be the use of a “side-by-side” file comparison utility which displays the team program output beside an “answer file” provided by the Judges ahead of time. This allows the judge to use the validator to examine the output and compare it with the correct answer, perhaps taking advantage of special capabilities of the validator program (for example, syntax-driven color highlighting of differences between the team program’s output and the correct answer file). When the judge is satisfied, the validator is terminated and the judge uses the PC² Judge client interface to enter a judgment.

An active validator is one which automatically examines the team program output and makes some determination of its correctness. The validator must contain program logic which directs how it determines correctness. This logic could be hard-coded within the validator (in which case the validator is almost always problem-specific), or could be more general (for example, instructions directing application of “difference testing” between the program output and a Judge’s “answer file”). In either case an active validator is one which is designed to make an “automated” determination of the correctness of the output of a team’s program.

For active validators, a set of conventions defined within PC² provide a mechanism for the validator to return to PC² an indication of what judgment it thinks should be applied to the run. However, such a validator response is never blindly accepted as the final determination of the judgment to be applied to a run. Rather, the validator output is displayed to the human judge as a “recommendation”. It is ultimately up to the human judge to accept the validator recommendation (or not). That is, it still requires the judge, at a minimum, to press a button to accept the validator recommendation.

E.2 Validator Configuration

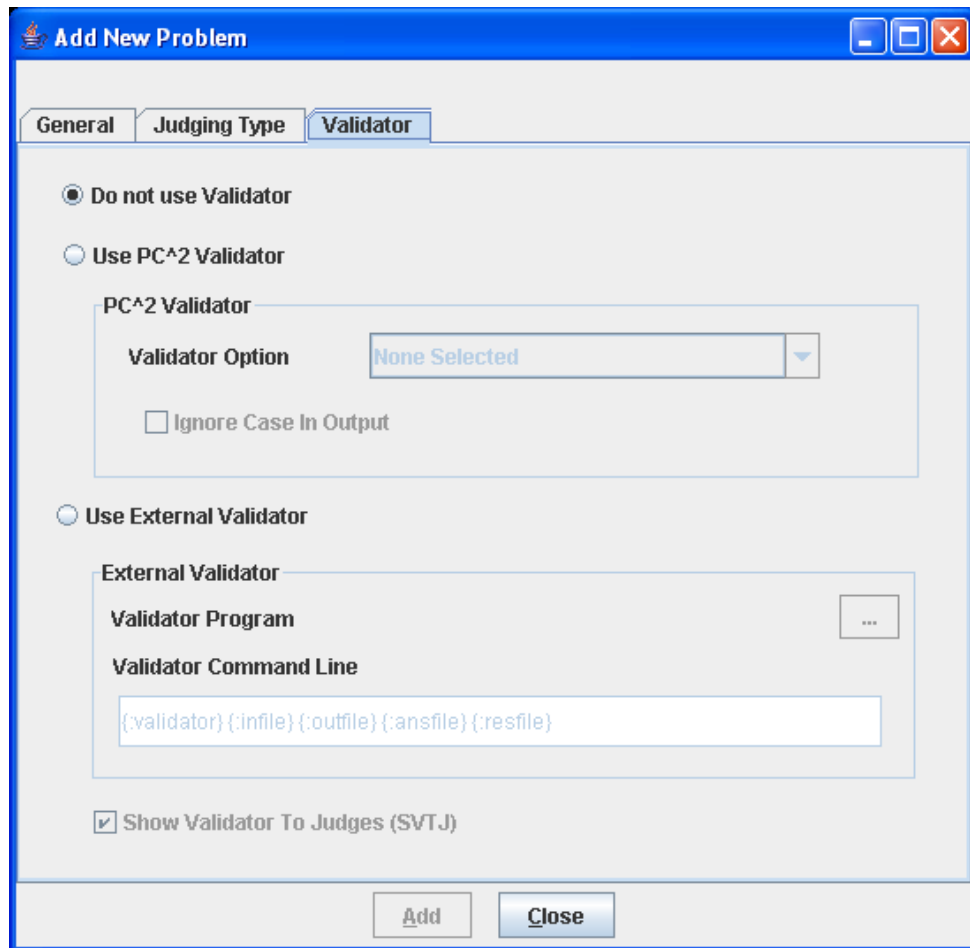
By default there is no validator attached to (associated with) a contest problem in PC². Validators can be attached to a problem by the Contest Administrator by using the **Validator** tab on the **Edit Problem** dialog. This displays the Validator configuration screen, as shown below. When the Contest Administrator configures a problem to use a validator, then when a Judge executes a

team program the specified validator will automatically be invoked as soon as the team program completes execution.

The Validator screen allows the Contest Administrator to choose one of two options for attaching a validator to the problem being defined: either the simple built-in PC² validator, or an arbitrary separate external program. If a separate external program is selected it can be either a standard system program (such as a “diff” utility or a split-screen file-compare utility) acting typically as a passive validator, or it can be a program written specifically for the validation of output from programs for this particular contest problem, i.e. an active validator.

The Contest Administrator can choose whether or not to display the validator result to the judge. Checking the box “Show Validation To Judges (SVTJ)?” when configuring a validator will cause the response returned by the validator to appear on the Judge’s display when the run finishes executing. If the checkbox is unchecked, the validator result will not be visible to the judge. The checkbox should normally be unchecked when using passive validators, which do not return meaningful result information (if SVTJ is checked and a passive validator is used, the response displayed to the judge will always be “Undetermined”).

The following sections describe the built-in PC² validator, the use of an existing external program as a validator, and the creation and use of a special-purpose problem-specific validator. For problem-specific validators, the ICPC standard for interfacing such validators to a contest control system such as PC² is described.



E.3 PC² Built-In Validator

PC² contains a built-in validator which can be used to compare the output of a submitted program with the “answer file” provided by the judges (specified when a contest problem is configured). The built-in validator is essentially a *very simple* version of the standard Unix “diff” file comparison utility. Selecting the “**Use PC² Validator**” option enables a drop-down list showing the various comparison options, which are listed in the following table.

PC² Validator Option	Meaning
1	Perform a standard “diff”: a character by character comparison of files differences
2	Ignore whitespace at start of file
3	Ignore leading whitespace on lines
4	Ignore all whitespace on lines
5	Ignore empty lines

When using the PC² built-in validator it is important to understand the exact meaning of the options which specify combinations (options 6 and higher). In particular, “X then Y” means that the validator first checks the output using case X and *then* checks the output using case Y, independently. Note that this is *not* the same thing as performing a single test applying both conditions simultaneously. This can produce unexpected results.

For example, “check the output, ignoring white space at start of file” followed by “check the output, ignoring empty lines” is not the same as “check the output ignoring white space at start of file and also ignoring empty lines”; if there are empty lines in the output file but not in the answer file, then when it validates using the criteria “ignore white space at start of file” the output will FAIL due to the empty lines in the answer file. ***Be sure you understand the operation of the built-in validator before using it!***³⁸

The PC² built-in validator is an active validator – meaning that it automatically returns to the Judge a recommendation for the judgment which should be applied to the run whose output was diff'd against the answer file. If SVTJ was checked when the validator was attached to the problem, then the returned recommendation is displayed for the judge.

E.4 Using An Existing External Tool As A Validator

If the Contest Administrator selects **Use External Validator** on the Validator configuration screen, it tells PC² to use an external program as a validator for the problem being configured. In this case PC² needs to be given two sets of information regarding the external program which is to be used: (1) the name of the file containing the validator, and (2) the command sequence which PC² should use when invoking the validator.

To understand how this works, suppose it is desired to use an external program named “*gvim*” as a validator for a certain contest problem. Further, suppose *gvim* is a program which can be given an argument, “-d”, along with two additional arguments specifying file names, and that what it does in this case is display the contents of those two files side by side, highlighting the differences.³⁹

What the Contest Administrator wishes to do is have PC² invoke *gvim* every time a run for this particular problem is submitted and executed by a judge, and pass to *gvim* two files: (a) the output of the team’s program, and (b) the answer file (correct output) for the problem (provided ahead of time by the judges). This will allow a judge to compare the team’s output with the correct output, using the capabilities of the *gvim* program to assist them.

Since our hypothetical *gvim* program already exists and is not (likely) written to understand how to make programming contest-related decisions about the correctness (or not) of the differences between two files, we will use *gvim* as a passive validator (one which does not return a

³⁸ Yes, we know it’s a crummy implementation. More motivation for you to write your own validator, as described in the following sections.

³⁹ In fact *gvim* is a program which exists on many systems and operates exactly like this. However, for the sake of this example, consider it to represent any arbitrary program; you don’t need to know anything about *gvim* to understand the example.

value to PC²). In that case the Contest Administrator would *uncheck* the SVTJ box, and then would select **Use External Validator** on the Validator screen.

In the **Validator Program** textbox the Contest Administrator enters the file containing the *gvim* program. This might be something like

`/usr/local/bin/gvim`

or

`C:\Program Files\gvim\gvim.exe`

Use the “browse” button (the button showing “...”) to select the validator program file name. Typically this is an executable program, but it could be any file. A legitimate file must be selected even if this field is not actually used.

In the **Validator Command Line** textbox the Contest Administrator enters the command to be used by PC² to invoke the validator. This command can include *parameter substitutions* similar to those allowed when configuring languages. As when defining languages, parameter substitutions are indicated by a set of matching curly braces with a colon as the first character and containing a substitution keyword, for example, `{:infile}`. The following table shows the substitutions which will be performed by PC² prior to invoking the validator:

Keyword	Meaning
validator	Represents the file name given in the Validator Program box. Note that it is not a required that the “validator program” actually be an executable program; it could for example be a text file.
infile	Represents the problem data input file as configured in the problem.
outfile	Represents the output sent to <i>stdout</i> by the team program when it was executed by the judge.
ansfile	Represents the judge’s answer file as configured in problem.
resfile	Specifies the name of the file into which an active validator must place an XML representation of the judgment.

Thus to invoke *gvim* as described above the proper **Validator Command Line** entry would be:

`{:validator} -d {:outfile} {:ansfile}`

This would invoke the *gvim* program (`{:validator}`) with a “-d” argument, passing it the team program output (`{:outfile}`) and the judge’s correct-answer file (`{:ansfile}`).

Note that while the above example uses an existing program (*gvim*) as the “validator program”, it is not a requirement that this actually be an executable program. As an example, the “validator program” file might be a text file containing a set of “rules” describing how to determine the correctness of a team’s output, and a completely separate program could be invoked via the **Validator Command Line** entry, perhaps passing the “validator program” file as a parameter.⁴⁰

For example, suppose a contest problem required teams to write a program to generate output which conformed to a set of lexical rules. If there existed a program named “*analyze*” which performed lexical analysis of the contents of a file according to a set of rules specified in another file, then the Contest Administrator might create an appropriate set of rules (the rules to which the team program’s output must conform) in a file named **rules.dat**, select **rules.dat** as the “**Validator Program**” file, and then specify the following as the **Validator Command Line** entry:

```
analyze {:validator} {:outfile}
```

This would invoke *analyze* as the program to be executed during the validation step, passing it **rules.dat** and the team output file as input. (What *analyze* would do with this is left as an exercise for the reader – but see the following section.)

E.5 Implementing a Validator

The use of an existing system tool (e.g. *gvim*) as a validator has the drawback that such validators typically must be *passive*, since they normally have knowledge of neither how to make decisions regarding correctness of team program output nor how to return such decisions to PC² even if they did know how to make them. The built-in PC² validator can be used as an *active* validator since it knows how to make decisions and knows how to return them, but as mentioned previously it is an extremely weak implementation.

Thus, it is desirable to have a mechanism which allows the use of a special-purpose program written specifically to be an active validator. In order to support the use of such validators, it is necessary to define a set of standards for interfacing validators to the system. Because a validator is a problem-specific (not contest-specific or contest control-system specific) entity, ideally these standards should be uniform and general enough that a conforming validator (and its corresponding contest problem) could also be used in conjunction with contest control systems other than PC².

Under the auspices of the ICPC, the PC² development team has worked with a number of other contest system development teams to define such a standard. Known as the **ICPC Validator Interface Standard**, it specifies two interfaces: the invocation interface from a contest control system to a validator program, and the result interface from the validator back to the contest control system.⁴¹

⁴⁰ It is arguable that a better name for this feature would have been “**Validator File**” rather than “**Validator Program**”, since it does not have to be a program.

⁴¹ <http://www.ecs.csus.edu/pc2/doc/valistandard.html>

The following subsections give a brief summary of the interfaces defined by the standard,⁴² followed by a description of the PC² implementation of the standard. Since PC² complies with the standard, any validator written to comply with these interfaces can be used as an active validator for a problem in PC².

E.5.1 Invocation Interface

The standard mandates that the contest control system is responsible for invoking the validator and passing it at least four command line parameters, as follows:

parameter1: a string specifying the name of the input data file which was used to test the program whose results are being validated.

parameter2: a string specifying the name of the output file which was produced by the program being validated when it was run using the data file named in parameter1 (that is, the name of the file containing the output to be 'validated').

parameter3: a string specifying the name of an arbitrary "answer file" which acts as input to the validator program. The answer file may, but is not necessarily required to, contain the "correct answer" for the problem. For example, it might contain the output which was produced by a "Judge's Solution" for the problem when run with the data file named in parameter1 as input. Alternatively, the "answer file" might contain information, in arbitrary format, which instructs the validator in some way about how to accomplish its task.

parameter4: a string which specifies the name of the "result file" which the validator must produce. The content of the result file produced by the validator is defined in the following section.

The requirements for passing parameters to a validator can be met by the Contest Administrator in PC² through the use respectively of the **{:infile}**, **{:outfile}**, **{:ansfile}**, and **{:resfile}** command substitution parameters in the **Validator Command Line**; PC² will automatically insert the appropriate values for the problem when the validator is invoked. Also, as required by the standard, the data file, program output file, and the answer file are in the current directory when the validator program is run. These conditions taken together specify how an arbitrary validator program should expect to be invoked by PC².

The standard specifies that the contest control system may pass additional command line parameters to a validator, as long as the first four command line parameters are specified as listed above, and that the interpretation of any such parameters is up to the validator. The Contest Administrator in PC² can pass arbitrary additional parameters to the validator by including them in the **Validator Command Line** after the four required parameters.

⁴² The interface descriptions contained here are necessarily terse; see the standard for complete details.

E.5.2 Result Interface

The standard requires that the validator result be returned in the “result file” whose name is specified by **parameter4** (above), and that the contents of the result file must be a valid “XML Document”. This means that it must start with a valid XML declaration,⁴³ such as

```
<?xml version="1.0"?>
```

The root element of the XML document must be of the form

```
<result outcome = "string1"> string2 </result>
```

The tag name “**result**” is fixed and required by the standard, as is the attribute name “**outcome**”.

“**string1**” is an “outcome string” defining the result (outcome) which the validator is reporting to the contest control system. If the value of “**string1**” is “accepted” (or any case-variation of that word), the standard specifies that the validator is indicating that the program output file “passed” the validation test(s). If “**string1**” contains any value other than a form of the word “accepted”, the standard specifies that the validator is indicating that the program output file “failed” the validation test(s).

In PC², the appearance of any form of the word “accepted” in the “**string1**” attribute of the result element in the result file causes PC² to assign a recommendation of “YES” to the run being executed. Recommendations are displayed to the judge if the SVTJ checkbox has been checked when the validator is configured with the contest problem.

If the value of “**string1**” is not some form of the word “accepted”, then PC² compares the actual string value to the “reject messages” originally defined in the **reject.ini** file. If “**string1**” matches one of the reject messages, then PC² assigns that message as the recommendation for the run being executed; otherwise, it assigns a recommendation of “Undetermined” to the run.⁴⁴

“**string2**” is an arbitrary message string being returned from the validator to the contest control system. The standard specifies that the interpretation of this string is up to the contest control system. PC² does not use the “**string2**” parameter from the result file.

E.5.3 PC² Extensions

The standard specifies that the XML **<result>** element produced by the validator may include other attributes in addition to the “outcome” attribute, and may also include additional

⁴³ Strictly speaking, the XML standard does not require that a document contain an XML header to be a valid XML document. However, the current PC² implementation expects a validator result file to have an XML header.

⁴⁴ Note: as mentioned previously, PC² appends the five characters “No – ” (that is, the word “No” followed by a space, a hyphen, and another space) to each “reject” message found in the **reject.ini** file. Therefore the strings contained in the “**string1**” attribute returned by a validator should also contain those characters.

(nested) elements; it also specifies that the interpretation of any such additional attributes and/or elements is up to the contest control system. Such additional attributes can be used to implement a variety of features.

PC² makes use of additional attributes to implement a form of security. Specifically, it expects the validator to define an additional attribute named “**security**” and to return in that attribute *the name of the result file*. That is, PC² expects the XML result file to look like:

```
<?xml version="1.0"?>
<result outcome="string1" security="resfile"> string2 </result>
```

where “**resfile**” is the value which was passed to the validator as the name of the file into which the results should be placed (and where **string1** and **string2** are as described above).

Each time PC² invokes a validator it generates a unique random name for the result file. When the validator returns PC² examines the contents of the result file and verifies that the **security** attribute value matches the file name. Since a user (team) program cannot know ahead of time what result file name PC² will generate, it is not possible for a user program to generate a “fake” result file which somehow gets used in place of one generated by the validator. While this is not a complete guarantee of security, it does make it much more difficult for a user program to circumvent the operation of the validator.

Appendix F – Language Definitions

As described earlier in this manual, PC² must be given a “language definition” for each language to be used in the contest (that is, for each tool which teams can use to write and submit programs). The language definition consists of four distinct text strings: the “Display Name”, the “Compile Command Line”, the “Executable Filename” specification, and the “Program Execution Command Line”.

In order to help in understanding how such language definitions work (and so that you will be better able to develop your own language definitions), it is useful to understand what it is that PC² *does* with a language definition. Language definitions are used by PC² in two circumstances: when a Team invokes a **TEST RUN** operation, and when a Judge or an Admin invokes an **EXECUTE** operation. The following algorithm describes the sequence of steps which PC² follows when either the **TEST RUN** button on the Team, or the **EXECUTE** button on the Judge or Admin is pressed.

1. The entire contents of the “**execute**”⁴⁵ directory (beneath the \$PC2HOME directory) are deleted. If something prohibits this clearing, the system stops and displays a warning message, and all remaining steps are skipped.
2. The submitted files are copied to the **execute** directory.
3. If the file whose name is specified as the “Executable Filename” in the language definition exists in the **execute** directory, it is deleted. This prohibits a team from submitting an executable file (or more correctly, they can submit it but it will never be executed).
4. The command specified as the “Compile Command” in the language definition is executed, using appropriate command parameter substitutions as defined earlier in this manual.
5. PC² checks for the existence in the **execute** directory of a file whose name matches the specified “Executable Filename”. If this file exists, it must have been created by the execution of the “Compile Command”. (This is how PC² determines whether compilation was successful.)
6. If the specified “Executable Filename” exists (hence, the “Compile Command” was successful), then the following operations are performed:
 - a. The data file associated with the problem (if any) is copied into the **execute** directory.

⁴⁵ In Version 9 the execute directory name is based on the logged in user, for example team 3 site 1 execute directory is named: `executesite1team3` judge 3 site 2 would be named `executesite2judge3`.

- b. The command specified as the “Program Execution Command Line” is executed, using appropriate command parameter substitutions as defined earlier in this manual.
- c. If this is an EXECUTE operation on a Judge or Admin (as opposed to a TEST RUN on a Team), then if there was a “Validator” associated with the problem, then the following operations are performed:
 - i. The “answer file” associated with the problem (if any) is copied into the **execute** directory.
 - ii. The command specified as the “Validator Command Line” is executed, using validator command parameter substitutions as defined earlier in this manual (see the Appendix on Validators).
 - iii. If “Show Validator Result To Judge” (SVTJ) was checked when the Validator was associated with the problem, PC² reads the result file created by the Validator (see the Appendix on Validators) and displays the appropriate result for the Judge.

Some examples of PC² language definitions which have been used in past contests and were known to work in those environments are shown below. Each definition consists of four lines, corresponding to the four text field entries required on the Edit Language screen when defining a new language under the main Administrator screen.

No guarantee is made that these definitions will work in *your* environment, nor that they will not become obsolete due to changes made by the various language tool vendors. All we can tell you is that all of these language definitions have been used successfully in past contests. Use them at your own risk.

Language: Java

```
Java
javac {:mainfile}
{:basename}.class
java {:basename}
Java
```

Language: GNU C++

```
GNU C++ (Unix / Windows)
g++ -lm -o {:basename}.exe {:mainfile}
{:basename}.exe
.\{:basename}.exe
GNU C++
```

Language: GNU C

```
GNU C (Unix / Windows)
gcc -lm -o {:basename}.exe {:mainfile}
{:basename}.exe
.\{:basename}.exe
GNU C
```

Language: Perl

```
Perl
compilePerl {:mainfile}
OK
perl {:mainfile}
Perl
```

Language: Microsoft C++

```
Microsoft C++
cl.exe {:mainfile}
{:basename}.exe
.\{:basename}.exe
Microsoft C++
```

Language: Kylix Delphi

```
Kylix Delphi
dcc {:mainfile}
{:basename}
.\{:basename}
Kylix Delphi
```

Language: Kylix C++

```
Kylix C++
bc++ -A {:mainfile}
{:basename}
.\{:basename}
Kylix C++
```

Language: Free Pascal

```
Free Pascal
fpc {:mainfile}
{:basename}
.\{:basename}
Free Pascal
```

One of the ramifications of the sequence of language-handling steps described above is that a team cannot submit a program whose file name is the same as the “Executable Filename” specified in the corresponding language definition. For example, if the Contest Administrator configured a language by saying that the result of a compile operation for the language was to produce an executable file whose name was always “a.out”, then if a team submitted a *source code program* in a file named “a.out”, then the source code program file would get deleted (step 3) prior to the compile step.

Normally this difficulty is eliminated through the use of command parameter substitutions; the Contest Administrator would not normally specify “a.out” as the expected executable file to

be generated by the compilation steps, but rather would use a specification such as “{:basename}.out”, and further a team would normally submit source code in a file named, e.g. “a.c” rather than “a.out”.

However, there is one scenario under which the mechanics of language handling by PC² can cause difficulties (or at least, confusion). This is the case of purely interpreted languages, such as Perl or shell-script. In these cases there is no “compilation” step which is expected to generate an “executable” file; the “source file” is effectively the same as the “executable” file (in the sense that the source file undergoes no transformation prior to invoking “execution”, since “execution” involves running an interpreter against the original source program).

For example, in the case of Perl, the Contest Administrator might attempt to configure the language definition as:

```
Perl
/bin/perl -c {:mainfile}
{:mainfile}
/bin/perl {:mainfile}
```

This definition says that the language Display Name is “Perl”; that the “Compile Command” invokes `/bin/perl` (the Perl interpreter) with the “-c” (check syntax) argument and the submitted file, that the result of “compilation” is to produce an “executable” file whose name is the same as the submitted file, and that following the “compilation” step PC² should check for the existence of the submitted file and if present it should invoke `/bin/perl` again, this time executing the Perl commands in the submitted file.

However, this language definition will not work, because of the steps which PC² follows: it will delete the submitted source code (`.pl`) file prior to invoking the compilation command. Again, the reason for this is that it checks for the existence of the specified “executable file” *after* the compilation step, and assumes that if the file exists then the compilation was successful. Thus if a team submitted a source file named “myFile.pl”, since the submitted file is the “executable file” which would be input to the Perl interpreter, the “executable filename” (after command parameter substitution) would also be “myFile.pl” – but the file would have been deleted.

It is still possible to use such languages with PC². The trick is to create a separate “script” file which acts as the “Compile Command” and has the effect of creating a separate file which has the same name as that specified for the “Executable Filename” and which PC² can test for after compilation and prior to invoking program execution.

For example, suppose the Contest Administrator creates a shell script file named “compilePerl” with the following contents (the example presumes a Unix-like environment, but a similar approach can be taken in a Windows system):

```
#!/bin/csh
perl -c $*
if ($? == 0) then
    touch OK
endif
```

This script basically says: run the “C-Shell” interpreter (line 1); have it execute the perl interpreter and perform a syntax check (`-c`) on the arguments passed to the script (`$*`) (line 2); check the system “status variable” (`$?`) and if it is zero (meaning no errors occurred) (line 3) then create a file named “**OK**” (line 4).

With this `compilePerl` script accessible via the PATH variable, the following language definition will allow a Perl program to be submitted and processed by PC²:

```
Perl
compilePerl {:mainfile}
OK
perl {:mainfile}
```

This language definition will invoke the `compilePerl` script telling it to syntax-check the submitted program file, then if the file “OK” exists (which will only happen if the Perl syntax-check was successful) it will invoke the Perl interpreter to execute the submitted program. Note that while this example is for Perl, other languages such as Bourne Shell (and other shells), Python, Ruby, and ‘awk’ can also use a similar solution.

The above example should provide some insight into the types of operations which the Contest Administrator can invoke from PC². For example, it is possible to create a script file which is invoked for the “Program Execution Command” and does any desired operation, such as copying a data file into the `execute`⁴⁶ directory prior to running the intended program. Basically any desired operation can be performed at either the “compile” or “execute” step, as long as one has a clear understanding of the PC² language processing algorithm described above. This organization of language processing gives a great deal of flexibility to the Contest Administrator.

⁴⁶ Note that the execute directory is now named: `execute <site><site#><loginname>` for example Site 1 Team 3 would create output in `executesite1team3`.

Appendix G – Using the PC² API

While the client interfaces (Admin, Judge, Team, and Board) in PC² are intended to be as general as possible, there may be situations where users would like a client to operate differently. For example, a user may wish to create a scoreboard that uses a different scoring algorithm, or to create a different sort of contest system interface for Teams. To support this, PC² provides a mechanism for users to create their own “custom clients” which interface with the rest of the PC² system.

The PC² API Java doc is in the distribution under `doc/api/index.html`. In the API Java doc there are code snippets which show how to use the API.

To use the API you must add the `pc2.jar` in the CLASSPATH (or build path).

The API in version 9.2 on has been used to implement a scoreboard, and read-only access to the contest data is implemented.

Use the `ServerConnection` class to connect to the PC² server. Here is the code snippet from the `ServerConnection` Java doc that shows how to connect to the server and access the contest data.

```
String login = "team4";
String password = "team4";
try {
    ServerConnection serverConnection = new ServerConnection();
    IContest contest = serverConnection.login(login, password);
    //... code here to invoke methods in "contest";
    serverConnection.logoff();
} catch (LoginFailureException e) {
    System.out.println("Could not login because " + e.getMessage());
}
```

After a successful connection, the `IContest` instance can be used to access the contest data/information.

Appendix H – Frequently Asked Questions

This appendix covers some of the common questions we get. For additional information, visit the PC² web page FAQ at <http://www.ecs.csus.edu/pc2/doc/faq/index.html>.

PC² FAQ and Troubleshooting Guide

This page lists some of the common questions which users have asked while trying to set up and use PC². It explains some reasons why problems occur, and provides suggestions for trying to eliminate them.

10.3 Version 9 Questions

10.3.1 Resetting A Contest

- Q: [I want to run a Practice contest, then quickly switch to a Real contest. How can I do this in Version 9?](#)
 - Q: [I'm using Version 9.1 \(or 9.0\) and I can't find the "Profile" function described above to accomplish a quick switch between a Practice and Real contest. How can I do the equivalent operation in V9.1?](#)
-

10.3.2 Connectivity

- Q: [When logging in, I get a message that says: "Unable to contact server, contact staff". What might cause this?](#)
-

10.3.3 Documentation

- Q: [Where is there more information about Version 9?](#)
 - Q: [How can I get the Java source code for PC² ?](#)
-

10.4 [Version 8 Questions](#)

10.4.1 [Java](#)

- Q: [What version of Java should I be using to run PC²?](#)
 - Q: [What can we do if we get the message 'OutOfMemoryError' while running the PC² Server?](#)
 - Q: [What does it mean when the system generates the message "NoClassDefFound" and won't run ?](#)
-

10.4.2 [Contest Configuration](#)

- Q: [Can you provide samples showing how to configure various languages for use in PC²?](#)
 - Q: [Is there a way to prepare PC² "Account Information" \(such as Team Names and Passwords\) ahead of time and then load them into the system?](#)
 - Q: [Is there a way to download and import team registration information from the ICPC site into PC²?](#)
 - Q: [How can I run a "Practice Contest" and then "restart" a real contest?](#)
-

10.4.3 [PC² Scoreboard](#)

- Q: [Is there a way to change the Scoreboard format?](#)
 - Q: [The scoreboard updates its HTML files only every 3 minutes; can that be changed ?](#)
 - Q: [How do we get the PC² scoreboard to print judgements and send balloon notification e-mails ?](#)
-

10.4.4 [PC² Validators \(Automated Judging\)](#)

- Q: [Does PC² provide support for "Automated judging"?](#)
 - Q: [Where is the interface between a custom \(user-written\) validator and PC² documented?](#)
-

10.4.5 [Networking](#)

- Q: [If the PC² Server is shut down or dies for some reason, is it necessary for Clients \(Teams, Judges, etc.\) to log back in?](#)
 - Q: [Why can't my PC² Clients connect to the PC² Server when I run the server on my Linux machine? \(It works fine when the Server is running under Windows, even if the Clients are on Linux machines.\)](#)
-

10.4.6 [Microsoft Windows-Specific](#)

- Q: [After configuring PC² with a language under Windows, when I try to execute a program submitted using that language I get an error message saying something is wrong with "ntvdm". What could cause this?](#)
-

10.4.7 [Other PC² Questions](#)

- Q: [Sometimes when a PC² team executes a TEST run they get back no results in the team output window -- even though they can get the program to produce output by compiling it at the command line. Also I notice that in the command window there are error messages like "Bad File Descriptor" and "IOCollector Error" or other similar messages. We have also seen this happen on the PC² Judge when judging a submitted run. What might be causing this? \(p.s. This seems to happen more often when running under Unix/Linux.\)](#)
 - Q: [When configuring a problem to read the data input from stdin, sometimes the results are incorrect or the program hangs. What might be causing this?](#)
-

10.4.8 [Getting Help](#)

- **Q:** [What if my problem is not covered in the above list? Can you still help me?](#)
-

Q: I want to run a Practice contest, then quickly switch to a Real contest. How can I do this in Version 9?

A: Starting with Version 9.2, this functionality is supported through the use of "**profiles**". Whenever you enter contest description data (e.g. accounts, languages, etc.), you are specifying values in the "current contest profile". The Admin module has a function which allows creation of *new* profiles (contest descriptions), including the ability to "*clone*" a profile (i.e. make a copy of an existing contest description) and the ability to *switch* to a different profile (i.e. make some other profile become the current profile).

Thus, to easily run a Practice contest followed immediately by a Real contest, you create the Practice profile up through the point where it contains all the information which is to be shared in common with the Real contest (for example, all the team accounts, language descriptions, judgements, etc.), omitting those data which are practice- or real-contest specific (e.g. the problem sets). You then *clone* the Practice profile, creating a new "Real contest profile". Each profile can then be independently completed with its profile-specific data (the data for the Practice contest problems entered into the Practice profile; the data for the Real contest problems entered into the Real contest profile; etc.).

When a server is started, if there is more than one profile present in the system it asks which profile should be used. The Practice profile can be selected and the Practice contest run; when the Practice contest is done it is a one-button operation to switch to the already-configured "Real contest profile", effectively "resetting" the system from Practice contest to Real contest. (Note: selecting or switching to a profile requires entering a password; the password for each profile is specified by the contest administrator when the profile is first created.)

Q: I'm using Version 9.1 (or 9.0) and I can't find the "Profile" function described above to accomplish a quick switch between a Practice and Real contest. How can I do the equivalent operation in V9.1?

A: Unfortunately the "profile" function is not available in versions prior to 9.2. The following manual steps can be used to accomplish the equivalent of a "reset" or "switch profile" from a Practice contest to a Real contest in Versions 9.0 and 9.1 (note that these instructions assume a *single-site* contest):

1. On the PC² Server machine, create two directories called (for example) "real" and "practice".
2. Copy your pc2v9.ini file into the "real" directory.
3. Change into the "real" directory.
4. Start the PC² Server using the command "pc2server", then login to the Server.
5. In a different terminal window, change into the "real" directory and start a PC² Admin using the command "pc2admin", then login to the Admin.
6. Use the Admin to configure all of the contest configuration items *which will be the same for both the Real and the Practice contests*. Typically this might include specifying things like Accounts, Languages, Judgement response messages, AutoJudging configuration, Group assignments, and loading ICPC registration data, but would **not** include entry of contest problems or judge's data

files, since these are typically *different* between the Practice and Real contests. It might or might not include values on the **Notifications** and **Settings** tabs, depending on whether or not you want those values to be the same for the Real contest as for the Practice contest. The point of this step is to configure, in the Real contest, *only those items which are the same in both contests*.

7. Exit (shut down) both the PC² Admin and the PC² Server.
8. In the "real" directory, execute the command "pc2zip". This will create a directory named "archive", and will place into that directory a "zip" file containing the current contest configuration (that is, the Real contest configured only with those data which are the same as for the Practice contest). The name of the archive "zip" file includes a UTC timestamp to help identify it.
9. Copy the .zip file created in the preceding step from the archive directory (under the "real" directory) into the "practice" directory.
10. Still in the "real" directory, restart the PC² Server and then the PC² Admin, and finish configuring the Real contest (enter the Real Contest problem descriptions, along with any other data that is unique to the Real contest). When done, you might want to shut down the Admin and Server and execute another "pc2zip" command in the "real" directory to make an archive backup of the final Real Contest configuration (this is not required, but it might be helpful later and it will not affect the previously-created zip file since each archive file is uniquely named).
11. Change into the "practice" directory and unzip the copied .zip file. The effect of this step is that you now have in the "practice" directory a contest configuration which matches the saved Real Contest configuration -- that is, a configuration containing only those data which are the same between the Real and Practice contests.
12. In the "practice" directory, start the PC² Server and then the PC² Admin, and finish configuring the Practice contest (enter the Practice Contest problem descriptions, along with any other data that is unique to the Practice contest). When done, shut down the Admin and Server and execute a "pc2zip" command from within the "practice" directory to make an archive backup of the final Practice Contest configuration (again, this is not required but it might be useful later).

To run the Practice Contest, change to the "practice" directory and start the PC² Server and Admin; use the Admin **Times** tab to start the (Practice) contest running. To switch to the Real Contest, first use the **Times** tab to stop the Practice Contest. Next, shutdown all PC² modules, change to the "real" directory, and restart the PC² Server and Admin from within that directory. Finally, use the Admin **Times** tab to start the Real Contest.

Q: When logging in, I get a message that says: "*Unable to contact server, contact staff*". What might cause this?

A: This could be a network problem (firewall) or more commonly the pc2v9.ini file was either missing or the server= entry has not been updated to point to the actual server host and port.

In the log file logs/pc2-startup-0.log there is an entry that shows which host and port is being contacted, for example:

```
090809 145358.046|INFO|main|info|Contacting server at localhost:50002
```

This shows the default host name which is likely different than the host name that is being used.

Q: Where is there more information about Version 9 ?

A: Review the [PC² Documentation](#) and the [PC² Wiki](#).

Q: How can I get the Java source code for PC² ?

A: PC² is not an Open-Source project; unfortunately, the source code is not available publicly at this time. This may change in the future, but it is not likely to change any time very soon.

10.5 Java

Q: What version of Java should I be using to run PC²?

A: We currently recommend JDK 1.4.2_05. This is the version we have tested most extensively, including running multiple Regional Contests. Other versions may work, but we have not done extensive testing on them. (Note that earlier versions of Java contain known bugs which can adversely affect PC² operations.)

Q: What can we do if we get the message 'OutOfMemoryError' while running the PC² Server?

A: This is because the Java Virtual Machine is running out of "heap space". The default values for initial heap size and maximum heap size vary among different JVM implementations. However, there are two Java VM switches which can be used to control the heap size: -Xms, which sets the initial heap size, and -Xmx, which sets the maximum heap size. These switches can be added to the command line in the script which is used to start the PC² Server (the script name is either "pc2server" or "pc2server.bat", depending on your environment).

For example, to start the Java VM with 64M of initial heap space and a maximum of 256M of heap space, add the following in front of the "-D" option in the "java" command in the script:

```
java -Xms64M -Xmx256M -Djava.security.policy=all.policy ...
```

You will have to determine which sizes will work best for your configuration.

Q: What does it mean when the system generates the message "NoClassDefFound" and won't run ?

A: This is a Java error message meaning that it could not find one of the "classes" (programs) required to run the system. The missing classes could either be PC² modules or Java system modules. There are a number of possible causes for this error. Here are some things you can try:

- Verify that the PC² system was unzipped "including subdirectories" and "with case-sensitivity". PC² uses many Java "packages", which must be stored in the proper subdirectories beneath the PC² installation directory. Make sure the program which was used to "unzip" the distribution properly created and maintained the hierarchical directory structure and the "case" of filenames in the system.
 - Verify that the CLASSPATH environment variable includes the Java/lib and PC² installation directories.
-

10.6 Contest Configuration

Q: Can you describe the Edit Language fields used for configuring languages in PC²? Can you provide samples?

A: Given below are descriptions of the fields and variables used in defining a language in PC², followed by examples for several commonly-used languages.

Field Names:

- Display Name - name for language displayed for users
- Compile Cmd Line - command to compile source file
- Executable Filename - name of the resulting executable file (see examples below)
- Program Execution Command Line - command line to execute executable

Variables: these are filled in by PC² at run-time; for example, if the input (program) file name is: "hello.java", then:

- {:basename} = name of input file without directory and extension (hello)
- {:mainfile} = name of file with extension (hello.java)

Examples:

```
Display Name: GNU C (gcc)
Compile Cmd Line: gcc -lm -o {:basename} {:mainfile}
Executable Filename: {:basename}
Program Execution Command Line: {:basename}
```

```
Display Name: GNU C++
Compile Cmd Line: g++ -o {:basename} {:mainfile}
Executable Filename: {:basename}
Program Execution Command Line: {:basename}
```

```
Display Name: Java
Compile Cmd Line: javac {:mainfile}
Executable Filename: {:basename}.class
Program Execution Command Line: java {:basename}
```

```
Display Name: Kylix Delphi
Compile Cmd Line: dcc {:mainfile}
Executable Filename: {:basename}
Program Execution Command Line: {:basename}
```

```
Display Name: Kylix C++
Compile Cmd Line: bc++ -A {:mainfile}
Executable Filename: {:basename}
Program Execution Command Line: {:basename}
```

```
Display Name: Turbo Pascal (DOS)
Compile Cmd Line: tpx {:mainfile}
Executable Filename: {:basename}.exe
```

Q: Is there a way to prepare PC² "Account Information" (such as Team Names and Passwords) ahead of time and then load them into the system?

A: Yes. Account information prepared ahead of time can be loaded into PC² with the "Load" button on the Administrator. Loaded values overwrite any values that were in the system previously. The format of the load file is as follows. File lines starting with ! or # in the first column are ignored. Each non-comment line has FOUR fields, separated by a pipe (|). The fields are:

```
Account_number
Display_name
Active (either "true" or "false")
Password
```

For example:

```
1|Number 1|true|pass1
2|Team Number 2||pass33
```

An empty "active" field is considered false (so the example above will cause team 2 to become inactive). True and false are not case sensitive.

Note that there is no way to associate a "Region ID" with accounts loaded from a file; in a contest using Region ID's the ID values must be entered manually (unless you are also using the "Import ICPC Data" function, which is covered elsewhere in this FAQ).

See the section "User Accounts" in the PC² Administrator's Guide for further information on loading account data and on the "Import ICPC Data" function.

Q: Is there a way to download and import team registration information from the ICPC site into PC²?

A: Yes, assuming you have access to the ACM ICPC Registration data (typically only Regional Contest Directors (RCDs) or their designees have such access). To do this, login to the ACM/ICPC website using your Regional Contest Director's account. There is a button there to download a "PC² initialization data" file. Once you have downloaded this file, unzip it in a directory of your choice (this will extract several files). Then on the PC² Administrator, go to the "Accounts" screen and click the "Import ICPC Data" button. This will ask you to navigate to the unzip directory and select one of the extracted files. Once you do this, PC² will read the rest of the files automatically and load the ICPC data (such as the team names).

Note: it is important to create PC² accounts BEFORE importing ICPC data. The import function assigns imported account information to PC² accounts in the order in which the accounts are encountered in the import data files (and therefore the accounts must already exist in PC²).

If it is desired to associate ICPC teams with PC² accounts in a different order than they appear in the ICPC download file, one solution is to simply edit the file "PC2_Team.tab" by rearranging the records in the file to reflect the desired PC² team ordering. Alternatively you can copy the PC2_Team.tab file to a new file named "_PC2_Team.tab" and edit that file by adding an additional field to the beginning of each record containing the desired PC² Team ID. PC² will automatically check for the existence of the file "_PC2_Team.tab" during an Import operation, and will use that file (including the PC² Team IDs) if it is present.

See the section on "Importing ICPC Data" in the PC² Administrator's Guide for further information.

Q: How can I run a "Practice Contest" during which contestants get to login and use the system to see how it works, and then "restart" a real contest?

A: There are several ways to do this.

1. Configure and run the practice contest. After the practice contest is over, go to the "Time/Reset" tab on the PC² Admin and push the appropriate "Reset" button ("Reset Site" or "Reset All Sites"). This will clear the problem definitions, runs, and clarifications from the practice contest but retain all other contest configuration information. You will then have to add the problem definitions for the real contest.
2. Configure and run your practice contest, then when the practice is over kill all PC² clients and the server, then use the command "pc2reset" on the server machine (typed in the directory in which the server was running) to reset the contest database. This will make a backup (zip file) of the current contest state, then delete all contest configuration information (putting the system back to an initial installation configuration, except for the contents of .ini files). You can then restart the server, start a new Administrator, configure the real contest (including loading account, language, and problem information), and start the real contest.
3. First create a subdirectory (say, "real"), configure the *real* contest in that subdirectory (meaning, start the server in that directory, then start an Admin and configure the contest). Then shut down the Admin and the server, move to a *different* directory (say, "practice"), start the server and then an Admin, and configure the practice contest. You can then run the practice contest from that directory. When the practice is over, simply kill all the clients and the server, switch to the (pre-configured) "real" directory, and restart the server from that directory.
Note that this method has the advantage of isolating (preserving) the practice contest data in its entirety, and it allows for a very quick switch from practice to real contest (the real contest problems do not have to be configured between the practice and real contests, as they do after the other methods listed above). However, this method does have the disadvantage that no data from the practice is carried across to the real contest -- including team accounts and any changes a team may have made, say to their PC² password.

10.7 PC² Scoreboard

Q: I would like to be able to change the format of the Scoreboard -- I want to be able to put the data into my own web page for example. Is there a way to get to the Scoreboard data in text form or something like that?

A: Yes; something like that. When you run a PC² Scoreboard module, it automatically generates a collection of HTML files showing contest standings and submission data in several different formats. These HTML files are updated and placed in a directory name "html" under the \$PC2HOME directory (PC² installation directory) each time there is a change in the contest standings state (roughly, each time a submission is judged). This updating is done automatically, on the fly as the state changes.

There are currently FIVE different HTML files generated:

- **full.html** -- a full listing of all (active) teams by name, ranked by number of problems solved and then by "penalty points"
- **fullnums.html** -- same as above, except also includes the PC² "Team Number"
- **sumatt.html** -- a summary page showing, for each team and problem, how many submissions the team has made on that problem and whether or not they have solved the problem

- **sumtime.html** -- same as above, except that instead of simply displaying "Y" or "N" if the team has (or has not) solved the problem, displays the contest-relative TIME (in minutes) at which the teams solved the problem (or shows "--" if the problem hasn't been solved)
- **final.html** -- a standings page formatted in the style used by the ACM International Collegiate Programming Contest World Finals

You can therefore get to the "text data" from the scoreboard by picking it out of the HTML files, if that's what you want. However, since the generated files are full valid HTML, you can simply use them as-is, or include them as components in your own HTML files (as frame data, for example).

For example, when we run a contest under PC² (as if there was some other way...) we never run more than ONE Scoreboard module for the entire contest (even if it is a multi-site contest spread over multiple continents). Once we have one Scoreboard module running, we build an external "master" HTML file containing references to the PC² directory HTML files (for example, we insert the various PC² HTML files into frames in the master page, along with things like a header frame showing the contest title and graphics and so forth). We then write a script (batch file) to copy the master page to some publicly published location at regular intervals (e.g. once per minute).

We then tell people who want to see the "scoreboard" (for example, Teams, Judges, Spectators) to simply point their favorite browser to the "public" location. The PC² Scoreboard will update the PC² HTML files whenever the standings change; the script will copy the updated HTML files to the public location at the frequency we specify.

This approach also makes it easy to "freeze" the (public) scoreboard at some point near the end of the contest, if that's what is desired: simply kill the script which is doing the copying (but not the PC² Scoreboard module itself). The internal scoreboard will continue to update, while the public HTML scoreboard will "freeze" at the last displayed form.

Q: The scoreboard updates its HTML output files only every 3 minutes; can that be changed ?

A: Yes. An entry in the pc2v8.ini file can be used to change (or remove) that delay. The minimum delay is in specified in seconds; if the delay is set to zero then there is no delay. An example is shown below.

```
[client]
minSecsBetweenUpdate=0
```

Three Notes:

- If the contest time is stopped (like at the end of a contest) then there will be no delay in updating the html files, regardless of the delay setting..
- The scoreboard only reads/changes the .ini value when a "start" contest clock is clicked on the Admin.
- The scoreboard ignores the current delay limit if a Yes judgment is sent to the board; any Yes judgment causes an *immediate* update.

Q: I have heard that the PC² scoreboard can be configured to print judgment notifications and/or automatically send email messages when a team correctly solves a problem. We'd like to use this function to tell our Contest Runners to take a balloon to the team. How do we get PC² to print/send balloon messages/e-mails ?

A: [Click here](#) to see the [Balloon e-mail in PC² Web Page](#).

10.8 PC² Validators (Automated Judging)

Q: Does PC² provide support for "Automated Judging"?

A: Yes, in several ways. PC² allows the contest administrator to associate, with each contest problem, something called a "validator". A validator is simply a piece of software written to accept (as input) the output of a team's program and return as the validator output an indication of the correctness of the team's program.

PC² has a "built-in" validator which can be used for validation, or you can write your own "custom" validator and load it into PC when configuring the contest.

Validator results (indicators of team program correctness) can be utilized in two ways. In "Recommendation Mode", the result returned by a validator is simply displayed for the human judge to see; the judge can either accept the validator "recommendation", or can assign a different response to the team's program. In "AutoJudging" mode, the system automatically accepts the validator result as the definitive answer for judging the run, and automatically forwards that result to the team. This allows PC² to be set up in "fully-automated" mode requiring no human intervention in the judging process.

The details of using validators with PC² are documented in Appendix F in the Admin Guide.

Q: Where is the interface between a custom (user-written) validator and PC² documented?

A: The interface and implementation details are documented in the Admin manual in appendix F.

The interface conforms to the international standard which is documented here: [International Collegiate Programming Contest Validator Interface Standard](#)

Note that this standard allows for system-specific variations, and that PC² makes use of certain of these allowed variations. For example, PC² *requires* the use of the "security=key" option shown in the standard. Be sure to read the *entire* Appendix F to gain a clear understanding of how to write a validator for use with PC²

10.9 Networking

Q: If the PC² Server is shut down or dies for some reason, is it necessary for Clients (Teams, Judges, etc.) to log back in?

A: In the current version of PC², Yes. Although the state of the contest (runs submitted, problem names, etc.) is retained between executions of the Server, the state of remote connections is not retained.

Note also that if the reverse happens -- e.g. a Client gets killed or shut down improperly without executing a "log out" operation -- the system will think the Client is still "logged in" and will not allow the Client to reconnect when it starts back up -- it considers it a "duplicate login". To fix this, simply use the "Force Logout" function in the Admin to make the system drop the login connection to the dead Client, then have the Client log back in normally. The same is true if a Server dies in a multi-site contest; before the Server can "reconnect" to the system, it must be "forced off" by the Administrator. (We're working on improving this aspect of the system...)

Q: Why can't my PC² Clients connect to the PC² Server when I run the server on my Linux machine? (It works fine when the Server is running under Windows, even if the Clients are on Linux machines.)

A: We have seen this happen when a Linux machine has not been properly configured for a network environment. Specifically, Linux machines typically start up configured to report their "machine name" as "localhost". PC² queries the OS to obtain the machine name, and then maps this name to a corresponding IP address. The machine name "localhost" maps to the "loopback" IP address 127.0.0.1. This ends up being the IP address which PC² uses for network communication -- which means Clients will be unable to contact the Server (or vice-versa) because they will use their own local loopback channel instead of broadcasting over the network.

To check whether your Linux machine has this problem, type the command "hostname" (or "/bin/hostname") at a command prompt. If this returns "localhost", your OS is not properly configured for network operations. Note that both the PC² Server and the Client machines must be properly configured in order for PC² to work properly.

One way to fix this problem on most machines is to edit the file /etc/sysconfig/network and change the HOSTNAME entry to a host name which resolves to a non-local IP address, and then reboot the machine.

Another approach is to use the "hostname" command, run as root, to set the hostname to either a specific non-local IP address or to a hostname which resolves to a non-local IP address. One user with this problem wound up changing the hostname before starting PC², via a command line like:

```
hostname $(ifconfig eth0 | grep inet | sed 's/.*r:\| B.*//g')  
or (depending on your shell)  
hostname `ifconfig eth0 | grep inet | sed 's/.*r:\| B.*//g'`
```

which is basically getting the address from the eth0 interface and setting the hostname to that value.

In any case, be sure to coordinate any changes in network addresses with your local network administrator.

10.10 Microsoft Windows-Specific

Q: After configuring PC² with a language under Windows, when I try to execute a program submitted using that language I get an error message saying something is wrong with "ntvdm". What could cause this?

A: "NTVDM" is the Windows NT Virtual DOS Machine. It is a Windows subsystem that attempts to allow older "16-bit" applications such as Borland's "Turbo" compilers to run in the 32-bit Windows NT environment (which is the environment used in Windows 2000, Windows XP, and Windows Vista, not just under "Windows NT"). However, PC² does not support use of 16-bit applications. You will need to replace your old 16-bit language system with a 32-bit compiler.

10.11 Other Questions

Q: Sometimes when a PC² team executes a TEST run they get back no results in the team output window -- even though they can get the program to produce output by compiling it at the command line. Also I notice that in the command window there are error messages like "Bad File Descriptor" and "IOCollector Error" or other similar messages. We've also seen this happen on the PC² Judge when judging a submitted run. What might be causing this? (p.s. This seems to happen more often when running under Unix/Linux.)

A: We have noticed that under certain conditions the Java virtual machine which is acting as the interface between PC² and the host OS does not accurately handle the connection between PC² and the external processes being used to execute the team program. In fact, this can also happen when executing the

compiler command used to produce the team's executable program. The problem seems to occur more frequently under Unix/Linux simply because those OS's are faster in the way they handle spawning and handling of the I/O channels for processes.

We are currently investigating this issue and seeking a permanent solution. Meanwhile, we have discovered that a workaround appears to be simply slowing the external processes down under Unix/Linux so that the JVM has time to keep up with the external OS process management operations.

For example, we have successfully eliminated this problem in a large Regional Contest by defining each "compiler call" (language) in PC² so that instead of directly invoking the compiler, it invokes a short script (which we call "slowdown"), whose contents are simply:

```
#!/bin/sh
nice -19 $*
```

We then define the compiler execution command (using the PC² Languages tab in the Administrator) to be (for example for gcc):

```
slowdown gcc -o {:basename} {:mainfile}
```

We do a similar thing to invoke the executable program:

```
slowdown {:basename}
```

What this does is simply execute the "slowdown" script, passing it the rest of the compile (or execute) command as arguments; the script in turn runs the "nice" command which again in turn runs the command (arguments), but at a low process priority. Note that the "slowdown" script must appear somewhere in the "path"; we typically put it in the PC² installation directory.

We do note that one undesirable side effect of this two-level process structure (running a script to run the actual program) is that if the actual program hangs for any reason -- for example, a team submits a program with an infinite loop -- and then the Judge uses the "Terminate" button on the Execution Timer, what gets terminated is the external script process, not the actual team program process running "niced". This means that the Judge will need to use "ps" and "kill" ("TaskManager" for Windows types) to remove the looping process. We are working on a fix for this side effect.

Q: When configuring a problem to read the data input from stdin, sometimes the results are incorrect or the program hangs. What might be causing this?

A: This is a known Java bug in versions prior to 1.4.1_05. The fix for this bug is to use **Java version 1.4.1_05** or later. For more details see Bug 4843136 at the Sun developer site.

10.12 Getting Help

Q: What if my problem is not covered in the above list? Can you still help me?

A: Well, we will be happy to try (please keep in mind that this is a totally volunteer project on our part). The first thing we ask is that you make the effort to read the Administrator's Guide and see if you can find out from there what is going on. The Admin Guide is available on our web page, and it comes packaged with the system when you download it. If that doesn't do it...

Each of the PC² modules generates entries in a "log" file. The log file names correspond to the module -- team.log, judge.log, etc. Look through the log files and see if there are any obvious error messages -- this will frequently give you a clue as to what is happening.

If you still can't figure out what is happening, create a set of "PC2 Zip" files of the system and send us an email message (pc2@ecs.csus.edu) describing the problem and containing the PC2 Zip files as attachments and we'll see what we can do to help. To generate the PC2 Zip files,

1. Go to the PC² installation directory **on the PC² Server** in use when the problem occurred and type the command "pc2zip".
2. Go to **any other machine(s) which were also involved in the problem** (for example, if the problem was associated with a Judge then go to the Judge's machine) and type the same command ("pc2zip") in the PC² installation directory **on that machine**.
3. Go into the "archive/" directory **on each machine**, find the .zip file created by the pc2zip command (it will have the current date/time as part of the file name), and attach each .zip file to your email message.

Please note that unless we have zip files from the **Server** as well as **any other machine(s) associated with the problem**, it is very hard for us to give you any meaningful help.

Appendix I – PC² Distribution Contents

This appendix covers the contents of a PC² software distribution. Each distribution contains a single (base) directory which contains these directories and files.

Directories

Directory Name	Contains
bin	scripts to start PC ² modules
data	XML Stylesheets and other data files
doc	API and user documentation
lib	PC ² Java library (jars)
samps	sample files
samps/data/xsl	XSLT files
samps/scripts	compile and other scripts
samps/src	Samples in C, C++, Java, etc.
samps/web	web resources and scripts
samps/web/xsl	samples for group XSLT for HTML

Files

Filename	Description
README	Late breaking and important info
VERSION	Version information

Example of directories found in a PC² distribution

```
pc2-9.2
pc2-9.2/bin
pc2-9.2/data
pc2-9.2/data/xsl
pc2-9.2/doc
pc2-9.2/doc/api
pc2-9.2/doc/api/edu
pc2-9.2/doc/api/edu/csus
pc2-9.2/doc/api/edu/csus/ecs
pc2-9.2/doc/api/edu/csus/ecs/pc2
pc2-9.2/doc/api/edu/csus/ecs/pc2/api
pc2-9.2/doc/api/index-files
pc2-9.2/doc/api/resources
pc2-9.2/lib
pc2-9.2/samps
pc2-9.2/samps/data
pc2-9.2/samps/data/xsl
pc2-9.2/samps/scripts
pc2-9.2/samps/src
pc2-9.2/samps/web
pc2-9.2/samps/web/xsl
```

Appendix J – Log files

Log files are stored under the logs/ directory, In version 9.2 server log files are also stored under profile/<ProfileID>/logs.

There are 4 different log file types:

1. startup log files – logging information before a module is logged in
2. module log files – logging information for a logged in module/client, **typically these are the files to check for errors when running PC²**
3. evaluations/judgments log – on server only, one line per judgment, see <http://pc2.ecs.csus.edu/wiki/Evals.log> for more details.
4. security log files – logging security issues when they happen

Appendix K – Acronyms, Terms, Definitions

Term	Definitions
SVTJ	Show Validation To Judges – a Problem setting which indicates whether to show the validation results to Judges

Appendix L – Reports Program

The **pc2report** program can be used to produce stand-alone reports about the state of the system. This program must be run on the PC² server machine (i.e., the machine on which the pc2server program is run). Each report generated by **pc2report** is identical in output content and form to the reports created using the Admin Report Tab.

Examples

These examples use the contest password 'newpass'; replace 'newpass' with the contest password entered when the PC² server was initially started.⁴⁷

Show Fastest Solution Summary report

```
$ pc2report --contestPassword newpass 'Fastest Solution Summary'
```

Show Runs report

```
$ pc2report --contestPassword newpass Runs
```

Show Runs report, use the report number (15) instead of spelling out report name

```
$ pc2report --contestPassword newpass 15
```

Usage

```
Usage: [options] reportName|## [[reportName|##][...]]
```

```
--profile name - profile name, default uses current profile. name may be a ##  
from --listp listing
```

```
--contestPassword padd - password needed to decrypt pc2 data
```

```
--list - list names of reports (and the report numbers)
```

```
--dir name - alternate base directory name, by default uses profile dir  
name
```

```
--site ## - specify the site number
```

```
--listp - list all profile names with numbers
```

```
--noProfile - do not use profile directory use pre version 9.2 location
```

```
reportName - name of report to print (or report number)
```

```
## - number of report to print (numbers found using --list)
```

```
$ pc2reports --listp
```

```
1 - Id: Contest-1526060434834405723 description: Real Contest name: Contest
```

```
2 - Id: Contest 3--613094433664018852 description: Real Contest 3 name: Contest
```

```
3
```

```
Default name : Contest
```

```
Profile ID : Contest-1526060434834405723
```

```
Description : Real Contest
```

⁴⁷ See the earlier note regarding the use of the **-F** option to avoid putting plain-text passwords on the command line.

Path : profiles\Pdf812e23-4234-46ee-ad3c-4011c8cb885e

Each of these will print the same report:

```
$ pc2report --contestPassword newpass --profile Contest 3--613094433664018852  
'Fastest Solution Summary'
```

```
$ pc2report --contestPassword newpass --profile 2 'Fastest Solution Summary'
```

```
$ pc2report --contestPassword newpass --profile Contest 3--613094433664018852 9
```

```
$ pc2report --contestPassword newpass --profile 2 9
```

Precedence for directory: --dir, --profile, then default profile dir

Version 9.2 20110720 (Wednesday, July 20th 2011 16:30 UTC) Java ver 1.6.0_22

build 2302 **Windows 7 6.1 (x86)**

List all reports available

```
$ pc2report --list
```

```
Report 1 Accounts  
Report 2 Balloons Summary  
Report 3 All Reports  
Report 4 Contest Settings  
Report 5 Contest XML  
Report 6 Contest Analysis  
Report 7 Solutions By Problem  
Report 8 Submissions by Language  
Report 9 Fastest Solutions Summary  
Report 10 Fastest Solutions Per Problem  
Report 11 Standings XML  
Report 12 Logins  
Report 13 Profiles  
Report 14 Plugins  
Report 15 Runs  
Report 16 Clarifications  
Report 17 Problems  
Report 18 Languages  
Report 19 Judgements  
Report 20 Runs grouped by team  
Report 21 Notification Settings  
Report 22 Client Settings  
Report 23 Groups  
Report 24 Evaluations  
Report 25 Runs (Version 8 content and format)  
Report 26 Run 5 field  
Report 27 Account Permissions Report  
Report 28 Balloons Delivery  
Report 29 Extract Replay Runs  
Report 30 Run Notifications Sent  
Report 31 Judgement Notifications  
Report 32 Active Profile Clone Settings  
Report 33 Sites  
Report 34 Event Feed XML  
Report 35 Notifications XML  
Report 36 Finalize-Certify  
Report 37 Internal Dump
```


Appendix L – Troubleshooting / Getting Help

Before getting help from the PC² Team

There are a number of documents and references that contain information about using PC², take the time and search these references before sending an email to the PC² team.

1. Search this document for an answer
2. Search the PC² Wiki or use Google to search for answers

http://pc2.ecs.csus.edu/wiki/Main_Page

3. Search the on-line FAQ

<http://pc2.ecs.csus.edu/doc/faq/>

4. Search PC² Bugzilla

<http://pc2.ecs.csus.edu/bugzilla/>

Getting help from the PC² Team

If you can not find an answer to your question, send the PC² team an email at pc2@ecs.csus.edu.

If you have attempted to use the PC² system and are having a problem, please email us a **pc2zip** file. This file is created by using the **pc2zip** script to create a special .zip file in the archive directory.