CLOWDR Conference Organizer's Manual

CLOWDR is a community-driven effort to create a new platform to support Conferences Located Online With Digital Resources. (A "clowder" is also a group of cats.) CLOWDR is created by Jonathan Bell, Crista Lopes and Benjamin Pierce. If you are interested in helping develop CLOWDR or using it for your live event, please email us.

This document is a (currently very incomplete) manual-in-progress for "operators" (conference organizers and their technical staff) of the CLOWDR conference platform. It assumes that you're already familiar with the user-level view of the platform, as described in the User Manual.

Help us improve this document by adding your own instructions, hints, suggestions, and questions! (Please go ahead and add material directly to the document rather than making comments in the margin, whenever as possible.) There is also a section for adding new questions at the end of the document.

Contents:

Getting Started

Resource Limits and Technical Constraints

Questions, Comments, Feature Proposals, etc.

Sections still to be written

Working Notes for ICFP

Getting Started

Request a Clowdr instance. To run a conference on Clowdr (or experiment with the platform to decide if you want to run a conference there), you first need to request creation of an instance of the platform. Do this by sending an email to hello@clowdr.org with your name, affiliation, and preferred email and the name of your conference.

Add organizers. After your Clowdr instance is created, you'll probably want to register other organizers so that they can use the system too.

- Go to the Registrations page under the Administration tab and enter information for each person. You can do this in two ways:
    a. Press the New Registration button and enter information for a single person.
    b. Press the Upload Multiple Registrations button and provide a CSV file in the following format:

```
name,email,affiliation,country
Jane Q. Smith,janeqsmith@janesmith.org,Jane Smith Ltd.,USA
```

The header line is mandatory. No spaces or quotes around data items. The affiliation and country fields are optional. The name field can be replaced by a pair of fields called ‘first’ and ‘last’.

- Make sure to press the Send button(s) that appear(s) in the users list. (You may have to Refresh your browser to see that the message has been sent.)
- Once they have confirmed their registration emails and created their accounts, you can make them admins by setting the appropriate switches in the Users tab under Administration.

Personalize the splash page: Navigate to the leftmost tab, click the Edit button at the top, and fill in appropriate details for the public view of your conference. Then go down to the bottom of the page, click the Edit button there, and fill in details for the registered participants’ view. Markdown syntax is supported.

Upload the program:

- First, create the program as shown in the `sample-program-upload.csv` file. Do not change the column headers or add new columns. Each row is an event. Events can occur multiple times, which allows you to schedule a presentation to happen multiple times in different timezones (events are grouped by title). Each event must be in a track (which categorizes in the program, you can use any names for the tracks that you like). The track is a logical grouping of events, and you can set a badge to be displayed in the program next to each event in that track (as part of the demo, I’ve done this for research papers -> “Talk” and Q&A -> Q&A; this is configured under Administration->Conference Tracks). Tracks are also used to organize the exhibit halls - if you tick the “Exhibit hall” option for the track, then the track will appear under the “Exhibit Hall” header, and arrange any uploaded posters for those events in a grid. Authors can upload their own posters, and admins can, too.
- Not all events need to have scheduled presentations (in the example, the posters are not scheduled, although they could be). If the event is scheduled (e.g. has a start/end time) then it should also be assigned a session and a room. The room represents a streaming source (YouTube, Zoom or iQIYI) that is used for a session, and the session represents a list of events that take place sequentially in the same room.
- Start/end times must be specified as YYYY/MM/DD HH:mm (as in the example). You can use any time zone that you like (as long as it is consistent for all times), and then when you upload the program (Administration -> Program), you should select the timezone that you are using in the upload. As you are experimenting, you should use the red “Delete entire program” to wipe out the program before you reload it.
- The second step (once you have the program loaded in) is to determine the streaming URLs and upload those separately (sample-room-upload.csv). This goes into Administration->Virtual Rooms.

Upload registrants: (As described above for organizers.)

Manually add last-minute registrants: If you want to add further registrants, use the same procedure as for organizers above.

Resource Limits and Technical Constraints

Technical constraints of the chat system. (Also see https://www.twilio.com/docs/chat/chat-limits) Each user can be a member of up to 1,000 channels, and each channel an have up to 1,000 users (we can also bridge multiple channels together in a way that does not guarantee that each user will see the exact same real-time ordering of messages, but will allow more than 1,000 users to see the same channel/same messages). Our UI design currently only allows for a single global channel (announcements), but we are overhauling the entire chat UI over the next week or two, and will support an arbitrary number of channels that conference organizers and/or participants can create (including a support channel).

Latency. You may see latency as a result of inefficient UI implementation, for instance: clicking on a paper to open a chat window might take a noticeable delay: this is because we wait to open the chat window until we have 1: asked the chat service to join you to the channel, 2: the chat service adds you to the channel, and 3: the chat service sends the channel information and messages back to your client. Other systems would open the chat window immediately and show a loading icon, progressively adding details as they arrive - this doesn't make them any faster, it just makes them appear faster. Most other operations within the app use data that is cached in your browser, and we are working to more aggressively pre-fetch data.

Limits on concurrent users / server resource requirements. We are using a shared cluster of servers (no server is dedicated to be only ours, but there are multiple containers in multiple datacenter on servers that run our app). The hosting plan that we use is the "Pay as you go" plan from Back4App (https://www.back4app.com/compare-all-plans), which guarantees that it can process 80 requests per second from client apps. Since the application runs in your browser, clicking around to different pages does not create new requests (the browser fetches data from the server in the background, and then when you click to different pages, it just presents different data that it already has fetched). Since the chat is processed in a different system (Twilio), the main operations that count towards these requests are updating the program, updating accounts, and creating/joining video rooms (there is some book-keeping that we do, roughly 1-2 requests against the backend for creating or joining each video room, but this is just when the user is entering the rooms and there are no load on our servers once a user is in a room). During ICSE we peaked at about 24 requests per second.

If you would like hosting beyond our shared plan, I would suggest considering Back4App's "Gold" dedicated server, which will provide you with your own set of dedicated servers for the job (do not consider the "silver" dedicated plan, which does not have the replicas, and is designed more for applications that just need a very large database rather than fault tolerance/many concurrent requests). If you are interested, you could open your own account with Back4App (who will also provide you with 24/7 hosting/infrastructure support directly), and I would be able to help you deploy Clowdr to that cluster.

Questions, Comments, Feature Proposals, etc.

If you have questions that are not answered in this document, please leave them here! If you know the answer to a question here, please answer it! If you have an idea, share it!

Sections still to be written

- Concepts and terminology:
    - How to talk about text vs. video chats
    - Papers vs. sessions
    - The “Lobby” (if we want to keep that concept at all)
    - How video spaces are grouped
    - The different kinds of twilio and zoom rooms
    - Authorization
- How to create a new Clowdr instance
- How to upload and manage the program
- How to upload and manage participant data
- Recommended flairs:
    - The hiring badge was a good idea - maybe that could have also been used as a way to indicate willing mentors?
- Operational advice for organizers -- e.g.
    - Schedule several training sessions on Clowdr (and record some of them)
    - Pay attention to informing and integrating newbies and students -- you are going to have a lot of them!
- Operating the system during the conference
    - The “Security Shield” feature
- Zoom licenses / JWT keys

Working Notes for ICFP

- Mapping from events to rooms needs to be doublechecked
    - NOTE: do we need to upgrade the zoom account? How many hosts?
- If the event is held in a zoom room, we need to add the host
- Can we set up pre-arranged text chat rooms, independent of events and video chats. If so, what should they be?
- We should have a carefully created set of preset video chat rooms
    - Information Desk
    - One persistent video room for each workshop so that they can use it to hold their social sessions. (Erlang specifically asked for this.)
    - Clowdr tech support (or should this be a text chat room?)