TWiki > DICE Web > LiveSupportChatService (11 May 2020, GrahamDutton)

Project 392: Live Chat Service

This page is the 'project homepage' for CompProj:392.

The established requirement is that we'd like some way to bridge

1. a "live chat" service, accessible from computing.help or similar user-facing web pages:
2. our established issue tracking / ticketing system, RT
3. our established chat service, using XMPP

The idea is that this would be used by users of informatics services to interact with CSOs and COs alike, providing another avenue to access computing support.

Phase 1a: Frontline Support Consultation

I wished to establish a sensible workflow for front-line support and hear their concerns as it might influence technical selections.

Phase 1b: Technical Options

In parallel with the above I investigated technical options as this would inform what sort of workflows are possible.

The outcome of phase 1 would be a report with recommendations for implementation. Prototype / mock-up implementation(s) might be available if easy to achieve.

Phase 2: Research / Progress Report

Planning

This was done towards the end of 2019. The consensus across front-line staff was that, as a new means of users reaching support, this would could quickly become unsustainable amidst all other methods of contact. However as a means of reaching out to users, a chat mechanism would be useful. There was strong support for the 'computing-initiated' model of chat from COs.

Following this consultation I'm proposing a model whereby COs can initiate a text chat with a given user which, on completion, dumps the contents of the chat into RT (ideally this would also support COs adding a summary of the conversation at the top of this 'dump' since chat logs are often necessarily long and sprawling).

My suggestion of a compromise method is to offer a live chat "facade" at a later date, which effectively initiates an RT ticket -- but creates the associated chat, should it later be useful to follow up. User expectation management would be extremely important for this to be of benefit.

Technical research revealed an embarrassment of options. Narrowing these down proves tricky. Almost all of the major extensible open chat systems support web interfaces, and possible solutions range in complexity to a Jabber "bot" running on / alongside our existing XMPP service, to integration with Microsoft Teams using its connectors to send transcripts.

Dozens of candidate packages were superficially evaluated, from 'web-first' solutions such as tawkto, to team chat (Slack-a-like) software such Zulip, Mattermost and Rocket chat), to true loosely-coupled federated setups such as Matrix/Riot and XMPP (of which, of course, we have the most experience). More recently the extensibility of our current pandemic team chat , Teams, was investigated. With no one technology identified yet, this is a good time to report on provisional requirements, and return to evaluate more carefully later with a series of trials and tests.

Requirements

There's an expectation of resource being limited a small collection of services running on a modest VM: a front end; a chat engine/database of some sort, and some RT integration scripts or plugins

We need to keep the service relatively low-maintenance. Cleanup should be automated, both chats and user records.

I'd expect only nominal storage requirements; active chats and data 'in motion' will be stored on that server, but the system should be designed to dump all completed / expired chats into RT and support no long-term storage.

For simplicity, the user front-end can sit behind Apache, or otherwise trivially integrate with our authentication system.

The CO-front-end can use the web and/or an existing chat client. I do not wish for this to require COs to use novel client software, however, so this limits us to web and/or XMPP (or something more exotic such as MS Teams, should the API prove sufficiently flexible!).

The system must be able to generate transcripts for transmission to RT. It doesn't need to support email if it can use RT's (Kerberised HTTP) API to create/update tickets, but email is probably by far the simplest.

The chat system should support minimal understanding of presence (i.e. is browser window open?) so that COs can abandon chats and move back to RT/email, and also for future use where end-users can be given some expectation as to whether their chat query is "live".

Any solution needs to be carefully evaluated with security / data integrity / protection in mind: as well as any security review of the self-contained service, we must evaluate the consequences of the integration with RT, since this system stores a considerable amount of personal information and even insert-only into the 'wrong' ticket has security and data protection implications.

Roadmap

I'd anticipate an incremental implementation; this help drive my fragmented development effort and also allows us to view the system better as series of interchangeable components, which means helps prevent 'buying into' one system until we've had a chance to evaluate it.

I'd expect to see these features in **roughly** this order:

1. Chat engine running
2. a. Web access to some sort of chat engine.
3. b. Authenticated access to said chat engine
4. c. Chat system should have some understanding of current RT ticket
5. d. **unauthenticated** access to said chat engine(?) What would be expected of this?
6. a. Transcripts can be sent to a given RT ticket by email
7. b. Transcripts can be sent to a brand-new RT ticket
8. c. Transcripts can be sent using the (Kerberised HTTPS) RT API rather than email
9. d. Transcripts permit 'summary' to be sent at the head of the chat, folding the rest of the detail.
10. a. COs can initiate a text chat with a given user, sharing chat link automatically.
11. b. Chat can be initiated from within a link in RT (like the 'Bugzilla' link)
12. c. COs can initiate a text chat with an unauthenticated link(?)

I say "roughly", as I'd anticipate going back and forth a little as we evaluate things. For example: authentication is mandatory, if but asymmetric authorisation (i.e. operators vs end-users) proves tricky, it could be simple to have the chat system email **every** transcript to RT's "new ticket please" system and reconcile these transcripts by hand, until a better system is developed.

Some optional extras:

1. Non-web client access at the CO side
2. a. It could be possible to initiate using only an RT#
3. b. This should extract ID details from the RT API
4. Support for integration into all of our RT systems - not just RT4

And, for the reverse, end-user-initiated live chat:

There's also lots of choice here in terms of pre-packed solutions There are lots of widgets available to plug into existing sites, but almost all seem oriented towards a 'sales' model where there's a near-endless supply of front line / reception staff to respond to initial queries and forward as appropriate. This doesn't suit our more resource-constrained system - but some aspects of these systems offer options such as chat-bot driven triage and searches of our documentation / status pages which could be attractive.

1. a. Integrate new ticket / live chat link into computing.help (opening a new chat using self-contained chat service)
2. b. "pop-up" option for chat (to appear ostensibly within computing.help)
3. b. Requires expectation management - presence information could be useful here
4. c. Chat-bot triage, searches computing-help before requesting precise description / new RT ticket.

Phase 3: Evaluation

The next step is to experiment more formally with some of the packages, and report on these trials with a view to implementing.

-- GrahamDutton

Topic revision: r5 - 11 May 2020 - 15:13:38 - GrahamDutton