# Roadmap



## Sections of Tim's presentation

Unrupt demo notes and Github readme page

1. Introduction
    - Start Introduction
    - Play Introduction

2. Podcall demo starts at 5:19 mins; explains rendezvous server
    - Start Podcall
    - Play Podcall

3. Unrupt demo
    - Start Unrupt demo
    - Play unrupt Demo

4. How it works; Webaudio issues and summary
    - Start How it works
    - Play How it works

5. Rest of session
    - Start Rest
    - Play Rest

Watch Tim Panton's full session at Kamailio World 2017 Thanks all:)

# Verbol Speak as peer to peer voicechat

Verbol Speak is an open source fork from our **unrupt demo** which re-implements in peer to peer form, the autochat feature of **Verbol voice Studio**. Speak adds voicechat and phone integration, using notifications between peers instead of servers being always there.

The Speak features deal with calls not being connected but a participant has a thought to say, then getting connected, and then not again and so on while continuing the conversation seamlessly. When we implement speak notifications, and out-buffering are added to unrupt features, conversations will be seamless and telephone tag a thing of the past.

For example if you want to say something to someone, just touch their icon and speak whatever is on your mind. They are notified that you are speaking to them and can choose to listen and reply live or later when convenient. If later it seems to them like live, so they can reply just by speaking while listening.

So while listening they can reply and the other party is notified and can likewise listen now or later. On each exchange either can add something to the conversation, so it keeps going and feels like a live call with breaks, without the parties having to be available at the same time.

## Visibility

For convenience we need to see when the far end is paused and how much catching up it has to do. The screen sharing built into WebRTC can be used as the default background in an audio call, so we see the other phone and its state just as its own user sees it. Screen sharing thus shows us the state of the far pause and its buffer indicators.

A limited form may be used during a video call to show just the state indicators. The user may wish an option to limit screen sharing to when the Speak webapp itself is being displayed. If not limited, then the speak conversation could continue and share screen while the user swaps to other browser tabs or apps subject to platform constraints.

## Screen sharing

Use the built in screen sharing of WebRTC as an additional option on the video on/off toggle. This will show the state of the far webapp including whether paused or muted an the buffering, and if not too hard on bandwidth should be the default display in the far video window when its video is off.

At the same time we do this we should offer the choice between front and rear cameras as well. So the video toggle when touched and held should expand to show the four options of video off, screen, front-camera, rear-camera. Or if screen share becomes the default display without video, the option of which camera can be in the menu and we retain the present simple video on/off button.

## Speaking-on-Mute warning

When the Mute button is on and its user is speaking, detect the voice and indicate to the speaker that he is still in Mute by flashing the Mute button and beeping.

## Connection data

In Verbol Speak we avoid gathering any data about the people using the devices for their conversations, to avoid any privacy obligations when we or an open source developer offer a Speak service. It is the connections between devices which our unrupt and speak technology make persistent, as opposed to saving personal data about people.

Product companies who we partner with already maintain personal and group association data and can utilise those for connections when implementing our technology in their products. We do not wish to compete with them by creating another silo of personal data.

## Make connecting easier

When displaying QR code also display the link and offer to copy to clipboard and confirm if selected.
Make connecting automatic if both webapps are up, and dont depend on displaying or closing the QR code.

There may be some simplifications we can make to the connection logic as per Tim Panton's podcast example and implementing notifications should improve things.

## connection naming

Speak should let you name the conversation designated by the unrupt token,
and persist it to the initiator's home screen from the start of the conversation.

When the gadget first loads it will ask its user to provide a name to describe the conversations he is the initiator of; this replaces the readme link which then is accessible from the menu.  Names can be nicknames, first names or made up.

Then when the user elects to start a new conversation, he is asked to name the other party.
We display the conversation name thereafter and access the "new" function from the menu.
The new also occurs on first loading of the webapp without an unrupt parameter.

So the title line when connected becomes the name given to the invitee followed by the name given to oneself.
Adding ones name should be invited but not required on loading and redoable any time just by clicking on it; likewise the name for the conversation.

The names in the title line can be used to add the descriptive text to a hyperlink for the conversation, so we add a copy to clipboard feature on every new conversation of the whole hyperlink as soon as the name is entered.

Q can we save to home screen programmatically?
If not can we retrieve the textual name from home screen created by user when he clicks the icon to load

NB the positioning of the 2 links in the title line needs reversing to correspond with the waveform and video in video position as far is left and near is right so showing the example assuming we reverse the sequence in the title line. There is the alternative of having near to the left which was my instinct; if we do that the hyperlinks will read Alice to Bob.

## example

**title line starts**
Speak Readme Menu

after loading and Alice has put in her name
Speak Alice Menu

after Alice has clicked speak, and perhaps spoken in a new call she names Bob
Bob Alice Menu

when Speak or the Menu is clicked to start a new call, the QR code is displayed,
and we should copy a full hyperlink to the clipboard Bob from Alice
to indicate this link is by Alice's webbapp inviting Bob.

or all from Alice's-work for team conversations, ideally with multiway calling,
until then multiway voicechat with two-way calling when any 2 participants are available.

Such links are suitable for saving by either participant
and its nice they show the conversations Alice invited,
and the conversations others invited with distinguished by the sequence of the names,
or by the keywords "from" and "to" which is safer.

We can choose github team and repository names to Speak and Invite and/or add to the link text.
and in display eg

Speak invitation to Bob from Alice

and its URL would be as follows, adding parameters for the name so it survives the URL being copied from browser address line.

https://speak.github.io/invitation?to=Bob&from=Alice&unrupt-id=xxxxx...

Idealy the user can have choice of near being to left as Brian wishes or to right as now; facebook and whatsapp let you move the local window anywhere inside the far one, can we do that in a webapp? If we switch around or let the user choose, these links would adapt and do need te keywords eg:

Speak invitation from Alice to Bob

## Pause parameter

To facilitate Verbol Speak we need to have a parameter on unrupt links:

Pause = on|off

When someone starts speaking and the webapp is not yet connected to a peer, it sends a web notification to the other party with the choice to accept the call to listen now or accept the call in pause mode to listen soon, or ignore.The first two choices generate a link with a different pause parameter as follows.

Webapp-URL/?unrupt=id-token to accept the call normally

Accepting a call in normal mode lets the acceptor hear anything that comes in as it arrives. Some could have come from the far outbuffer while the connection was established.

Webapp-URL/?unrupt=id-token&pause=on to accept the call in pause mode

Accepting a call in pause mode lets the acceptor finish something else while his in-buffer catches up with anything in the far's out-buffer and then anything else the other says before near unpauses to listen and reply.

## Speak logic

Connection is first established by Alice's webapp generating a link with a unique unrupt parameter that she shares with Bob who must click it while Alice's webapp is running. When loaded the webapps will find each other by opening a websocket with the same rendezvous server, which matches them

by the unrupt parameter, so they can send each other messages to coordinate.  All media is sent peer to peer via WebRTC.

At the time the initial websocket connection is made to the rendezvous server, the webapps will both register to receive notifications direct from each other if that is possible. If not we will upgrade the rendezvous server so it can pass on as notifications, messages received via websocket.

This means that if a webapp registers for notifications on first lading, then when a user device later deactivates the browser running the webapp, a service worker is left to respond to a push initiated by an active webapp to notify its user to wake up the other webapp.

We need to send notifications to the far party in a conversation when a webapp is loaded but the other is not which means the call is not connected, and also when a party to a conversation unpauses to start listening. Notifications when someone is first speaking after loading would be helpful but not on every utterance. Perhaps re-sent when [another] 1-x minute(s) of speech has accumulated in the outbuffer.

There may be a benefit for a notification subject to user control after the connection dropped during a conversation. A long period of silence such as a few minutes may trigger disconnection in some WebRTC clients.

## Walkthrough

Alice initiates a new unrupt connection and
starts speaking her thoughts to Bob and
shares a link with an unrupt parameter in it with Bob.
Until Bob accepts the invitation Alice's speech is held in Alice's out-buffer.

Bob clicks the speak link just shared by Alice
If he has a thought he can start speaking it and it will
be held in Bob's outbuffer until connection is established.

When connection is established anything in the outbuffer of either party is
sent from that party's out-buffer to the other webapp.

If either has a thought while listening the
incoming stream will be buffered in that party's in-buffer.

In case the connection drops because one party closes his webapp or phone.
Alice and Bob both save the link for their connection.

Later a party has more to say so clicks on the link again.
If the other party's webapp is still active they connect via rendezvous server.

If other party's webapp is inactive we send a notification.

eg If Bob has more to say he clicks the same link he originally sent to Alice to restart his webapp.
Alice is automatically notified that Bob's webapp is loaded to this connection,
because when one webapp of a speak link is activated it notifies the other webapp(s).

Alice has three choices to:
- accept and listen now
- accept in pause mode to listen later
- ignore

Alice can accept and listen now in which case she may have joined after Bob has started speaking,
but Bob's outbuffer will ensure Alice hears it all by buffering until Alice''s webapp loads and pulls
Bob's utterances from the outbuffer;

or Alice can accept in pause mode, so incoming media is stored in Alice's inbuffer
until Alice is ready to listen and so unpauses.

When Alice does later unpause to listen then,
- if the call has disconnected a notification is sent to Bob, and the process repeats
- if still connected the call continues as an unrupt call with autochat

## Multiple conversations in same browser

### Problem

The my-id which is the unique token that identifies an instance of the webapp
is at present stored in browser temporary storage and is lost when an instance is closed.

The unrupt-id for the webapp instance that initiated the conversation
is held in persistent local store in the browser,
so is retained when a webapp tab or the browser closes.

There is only one saved unrupt-id for the browser from the same host name,
so this limits conversations to one invitation per host name per browser,
and the most recently saved unrupt-id will come back to
any of that browser's instances whatever the unrupt-id on the link parameter is.

### Browser Instance Table

To handle this we can have a table of my-ids encountered by this browser
keyed by host/unrupt parameter and
saving the my-id already created in this browser instance
for the conversation hosted by the unrupt-id when first loaded
and then copying to the my-id to re-establish the same instance.

when reloaded for the same unrupt-id, and optionally any associated buffer contents.

Then when my-id is being set on loading, the table is matched to the unrupt id and
the matching result sets my-id for this webapp, and the saved data is copied back,
so this browser instance can be involved in more than one conversation
each from a different invitor instance with a different unrupt-id.

This should work even in different conversations initiated from the same host
in the same browser instance, as well as from other hosts,
but probably not for tabs for both the initiator instance and responder instance
in the same browser on the same device as this would put 2 entries in the table?
This might be resolved by the rvs sending a message to both
asking the one with the unrupt parameter in its local store to identify itself,
prior to attempting to connect.

## Persistent conversations

To enable persistent conversations on devices such as smartphones with battery concerns which cause browser tabs to sleep or close, we need a service worker to be set up by the webapp when it first loads, so this can be sent a push notification, so its user can wake up his webapp, when another user is speaking.
Service workers cannot access browser local storage but can access IndexedDB
as reported in this blog which also says

> "Scope is very important in service workers. Unlike web workers where one main thread can create unlimited web workers, only one service worker can exist per scope and their existence is independent from the main thread. Using this property, service workers can be accessed by many tabs in the browser at the same time."

This means in verbol speak that webapps loaded from the same hostname share the service worker for a particular browser instance. So it must be able to handle notifications from webapps in multiple conversations.

### Problem

When a webapp instance is loaded it sets up a websocket connection to the rendezvous server to send and receive messages from the other webapp to connect, and also to change pws state via rvs as now.

But if not connected, for instance because its device went to sleep,
and so the websocket to the rvs dropped,
then the rendezvous server doesn't know about it any more,
as this instance will no longer be in its table of websocket connections,
unless we somehow save a past instance's notification registration.

### Notification Questions

If an instance is not connected but has been before
Q can the service worker keep the web socket open?

> if so we dont need the push and can just send messages through the rendezvous server
> to the service worker which can display the notification to the user.

Answer from github

jakearchibald **commented on Nov 3, 2017**

> We can't keep websockets open while the user is off the site for battery and privacy reasons. Web push is a better solution here.

If not (this is advised) we need to send a push to the service worker to ask the other user to re-open an instance
by clicking on a link in the notification containing the unrupt parameter
and to do this we need to remember the service worker's notification registration

Q how do we remember a webapp instance's notification registration?
Q do we need to upgrade rendezvous server connection table to support Notifications by remembering the registration so we can push to it?
ie update the rvs so when an instance has a notification registration its entry in the connection table is modified to persist instances when their websocket disconnects.

Q Is notification registration added to the data in the present instance table alongside connected instances and we switch to using notifications when the web socket is not connected?
or can it all be done in webapp by remembering in the webapp and send it to the rendezvous server in a message?

NB after the initial invite is responded to, which comes from the first instance of the webapp for a new conversation, either party should be able to restart a conversation after it lapses by clicking on the same link with the unrupt parameter for the conversation. That should load the near webapp which causes a push to the service worker for the other webapp so it displays the same link for its user to load the webapp again.

To ensure there is only one attempt at a webrtc connection between the wbapps, when both apps are loading and have a websocket open to the rendezvous server, we can coordinate to ensure there is only one webrtc connection request made between the two parties. We can accomplish this by making the webrtc connection request always be from or back to the initiator instance which we can distinguish by its my-id being the same as the unrupt-id.

Q is it best to always make webrtc connections out from the initiator instance or in from the responder instances? Or should we consider making the connection request from the first to re-initiate a connection by notifying the other, or vice versa?

Q in periods of active asynchronous use, can we have the rvs send a notification to each party at the same time so they can click to activate synching or speaking, eg a designated check-in time every day in which a connection is set to be established regularly?

Look here on browser compatibility for web notifiations
https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API

### Buffering questions

During the notification registration callback the service worker script can connect to the webapp sending the notification to pre-cache anything needed by the notified user's webapp when it loads after the callback and before the service worker is activated. Should we use this to transmit anything in the outbuffer of far to in-buffer of near?

Q can Utterances be saved from the outbuffer?
    If not we can use the record feature
    but better to avoid it if possible as a new source of bugs and limitations.

Q should we turn recording on and off when voice detected like in asterisk?
    If we do can we feed into VvS and serve at quality?

When we save utterances from the outbuffer into persistent local storage they can be keyed from a playlist kept in this instances entry in the instance table.

# Other wish list items

## PBX features

It is possible to implement features of a PBX in the webapp to run in the browser so each participant can carry on the connection on multiple devices and extend the conversation to others. For instance enable call transfer so a participant can transfer the connection from his phone to his PC and back.

Implement 3-way callng by way of each webapp being a member of a chain or tree of other webapps. Recursive operation of the webapps so each user has a copy of a personal PBX is interesting and will lead to further innovations.

### Call transfer

When we implement transferring a call to a different device the new webapp instance registers for notifications.

The user should have choice of whether the instance in the old device continues to receive notifications as well; default is yes.

## Shared Whiteboard etc

A shared whiteboard is available now in Verbol voice Studio using the HTML5 canvas feature. We wish this to be migrated to Verbol Speak to operate peer to peer and be enhanced to mark up documents with scrolling. Other useful features include playlist and navigation bar, for extended conversations, with automated calls to transcription service. If we support text chat it should be in the same voicechat message stream as the

utterances. We can also consider Speak as a client webapp to Vv Studio, as an alternative to the present phono based webapp.

### Other parameters

In the links to set up a Verbol Speak call, we need a "PWS=on|off" parameter to say whether a call should start with pause-while-speaking on or not to indicate whether to be in unrupt or normal call made. A "record=on|off" indicates whether the buffers are to be discarded so the call is off the record or saved so the call is recorded. When a call is recorded, be able to "play [utterance1-utterance2]..." as implemented in Vv Studio. A parameter to automatically transcribe utterances and/or translate using a nominated service is also needed.

## Immutable conversations

Another intention is recording of critical conversations immutably in a blockchain. Conversations in some circumstances should be recorded and the recordings should be untamperable so storing them as immutable data would make sense. Family Systems technology facilitates this because the utterance generation creates discrete transactions of the conversational exchanges instead of continuous recordings.

Furthermore, the voicechat playlist contains data on how much of an utterance a speaker had heard before starting to speak, which pauses the listening, so replies can be inferred to precise context. By including the software footprint in the blockchain as it transacts the conversation, security will be enhanced. Each user experience will be different because of varying network delays and the variations in sequencing. So we envisage that every device records the exact experience of its user and shares it with the others who do likewise.

Thus all sides of a conversation are recorded and shared with all participants and blockchained immutably. Then if there are any misunderstandings the original experience of each can be recreated for all. Another benefit of each device recording its version of a conversation is that a higher fidelity podcast of the conversation can be created by merging the local recordings of each speaker, in which network packet and codec losses are eliminated.

## Verbol Speak by annual subscription

includes patent, quality controlled trademark and share-alike software license,
ideally with metered rendezvous server per customer to count down the pre-paid instances. Needs crowd funding to get to minimum viable product.

[pay $10 per year get 10 instances py]
OR $1 pm for 10 py
pay $100 py get 1000 py so 10c each
OR $10 pm for 1000 py
pay $1000 py get 100,000 py
pay $10,000 py get 10M per year

pay $100,000 py get 1B per year
pay $1M per year get 100B
pay $10M py and get 10T py [or unlimited for one corporation, cant be shared in a pool]

## Open Source Synergy

In my mind this will fund an MIT licensed (silent on patents) open source product offering free for non-commercial use or for Linux which will feed us with more patented inventions for commercial licensing so the process becomes continuing. We have a ready made case study that is:

- Simple and clean in ownership
- Ground breaking
- Long lived
- Able to evolve
- Can partner with open source for more inventions.

## Developers

Open source Developers that contribute to turn our demo into a product get a shared 10% stake in the commercial patent license revenues and trademarked product revenues

And therby form a group of stakeholdes all little guys who are damaged by big company abusing the payent law so form a class for actions in law, and each such licensor sets ots own price.

And we can offer compilations of multiple patent family licenses like music compilation albums from different inventors as the altetnative of big basket portfolios where the individual inventions grt lost.

Our case study can be an archetype from which we can build a template for a blockchain license for other small and micro enterprises and chsarge a 10% administration fee on licensing revenues paid out to licensor.

## Call it Open IP

like open source for patent and trademark licensing

All licensees have a stake so could be a class of victims of patent infringement or abuse of process by bigcos as the article describes

that also says the pto reviews have reduced patent values dramatically as we have experienced and any invention can be targeted for recurrent reviews

If licensees are all small or micro entities and all transactions are in an open blockchain which inventor and all licensees all share and at a reasonale price then they would be hard for bigcos to go against

Each patent owner sets his prices and we offer compilations like in music a blues compilation from different artists. Preview of Proprietor License perk.

**The Unrupt Patents and Patent Applications**
**Systems and methods for improving audio conferencing services; Family Systems Ltd.**
**As listed by IPwe.**

| | |
|---|---|
| US-9087521-B2 | Jul 21, 2015 |
| US-9538129-B2 | Jan 03, 2017 |
| EP-3017589-A1 | May 11, 2016 |
| US-20150012270-A1 | Jan 08, 2015 |
| US-20150312518-A1 | Oct 29, 2015 |
| US-20170236532-A1 | Aug 17, 2017 |

# Parallel activity

fit unrupt into an app to make calls like phone
add speak functioality
upgrade Vv Studio to wrk with new Chrome
use Vv Studio for project communcation and determine what to re-apply
establish software plan
feasibility study waearable personal server
follow up on automated licensing

# Resources

Family Systems is looking for help to develop the "unrupt" demo to a full verbol "speak" product in a business model that makes this synergistic with corporate patent licensing. We have autochat and other ideas prototyped in Verbol voice Studio to make spoken conversations persistent and seamless and eliminate telephone tag.

This was implemented client/server using recording in Vv Studio and we wish Verbol Speak to be implemented peer to peer and record or buffer depending on whether participants wish to be "on or off the record". Our **unrupt** ™ demo undoes disruption caused by interruption; so we can experience pausing a call or speaking while someone else is without missing anything. Here is our Unrupt demo invite.

Our intention is these unrupt and speak innovations are delivered just as if talking on a regular phone rather than dealing with messages or voicemails. Internet phones can also add synched visuals to "show and tell". Both features together mean we can free our minds by expressing every thought as we have it, with confidence that we will be heard and responded to. This means never having to wait for the other person to be available before speaking or to stop speaking so you can speak.

# Links

Family Systems Technology - 2018_Dec 1
Verbol Voice Studio
Unrupt demo
Github readme page
Github licence page

Roadmap
Partner Opportunity
draft Patent license blockchain ideas
Earlier patents to which Family Systems is licensed

Copy of Roadmap 2018 re pbx in browser