XirSys Beta Platform Documentation

Version 0.2

Version: 0.2
Updated: 25th February, 2015
Author: Lee Sylvester

Contents

Version: 0.2
Updated: 25th February, 2015
Author: Lee Sylvester

Disclaimer

By reading this document, you agree not to discuss its contents with persons or entities outside of your organisation.

The XirSys Beta is an experimental platform demonstrating capabilities and possibilities considered for inclusion within the XirSys Public platform. The XirSys Beta will contain bugs and incomplete features and will exhibit ongoing changes which may be added at short notice. The XirSys Beta should not be used for production applications, but will be sufficient for the development of applications speculated to be released once features have been ported over to the XirSys Public platform.

All features provided within the XirSys Beta will aim to complement those within the XirSys Public platform. Any changes provided will also accommodate backwards compatible equivalents where those features are already provided in the XirSys Public platform.

Changes

As we progress with the beta platform, new features will be added. This may require changes which break previous versions. Where changes affect the public platform, we will provide additional legacy hooks to ensure backward compatibility. However, this may not be immediately available in the beta platform.

All visible changes will be listed below:

v0.2b Wednesday, 25th February, 2015

Signalling: Heavily modified signalling protocol.
Signalling: Added publishing / subscribing capability for one to many.

Introduction

Welcome to the XirSys Beta platform. As a valued customer, we have selected you to try out our Beta. The XirSys Beta aims to be an ongoing experimental area providing access to new features considered for inclusion within the XirSys Public offering. It is our goal to utilise this platform to rapidly develop and introduce new features which we feel would be of benefit to our customers and the XirSys ecosystem.

It is our hope that, as well as the traditional questions posed by XirSys customers, such as *“How do I do* x *with XirSys?”*, that we should also receive some positive criticism; including suggestions for how we can improve on our API, idea’s for new features and feedback on the performance of our platform.

Finally, we have created a repository containing a set of examples that will be helpful when learning XirSys or if you simply wish to try out the XirSys suite of services. You can acquire this repository by visiting http://github.com/xirdev/xsdk.

The XirSys example repository will eventually become the location our client JavaScript SDK, and will provide a means to try out newly released features, so please do check back often.

XirSys REST API

The XirSys API is now mostly REST compatible. We have made reservations to deviate from the REST best practices where we feel it is necessary to do so or where we feel it is beneficial to our end users.

The existing public XirSys API is fully backward compatible, with the exception of the analytics based functions. We will, however, re-implement these soon.

Note: packets returned by the endpoints listed below are of the JSON format:

{“s” : “<status>”, “p” : “<path>”, “d” : “<data>”, “e” : “<error>”}

In XirSys, all server responses will be of a literal HTTP response 200. However, the virtual HTTP response is depicted by the “s” parameter. We opted for this route so that we can provide users with details about failures and discrepancies in a meaningful way, which is otherwise not possible with true REST.

Erroneous responses will not contain data. Therefore, by looking for a value other than null in the “e” parameter, client code can respond accordingly when things go wrong.

/signal/token

Method: GET (also: /getToken)

This call grabs a secure WebSocket token from XirSys, which ensures that WebSocket data usage is restricted to only the application's owner. The tokens themselves are encrypted, which is expected by the WebSocket endpoints and required to set up a successful connection. Invalid tokens or incomplete data within tokens result in an unsuccessful WebSocket connection. This is typically called whenever users have to interact with our signaling server.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< www.yourdomain.com >
application	< Your application >
room	< Your room >
username	< Connectee's identifier / username >
type	< "publish" | "subscribe" > (optional)

The type parameter is required for one-to-many calls. See the section on Publishing for more information.

/signal/list

Method: GET (also: /wsList)

This grabs a list of available XirSys messaging servers (usually just one) for signalling. No parameters are needed.

/ice

Method: GET (also: /getIceServers)

This request returns an ICE object of WebRTC STUN and TURN URL's, complete with credentials. Credentials are time sensitive and should be utilised within a 60 second period from requesting. This value can be modified by passing the timeout parameter.

If the passed secure parameter is set to 1, the returned ICE object will contain URL's protected by TLS (DTLS is not yet available for peers).

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< www.yourdomain.com >
application	< Your application >
room	< Your room >
secure	< 0 or 1 > (optional)
timeout	< number of seconds until expiry > (optional)

/subscribe

Method: GET

This is used to retrieve a broadcast URL for a published stream in the given room.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Top level domain >
application	< Parent application >
room	< Room to be created >

The returned URL can be assigned to an HTML5 video tag's src attribute for viewing of the stream.

/domain

Method: GET (also: /listDomains)

Returns a list of all domains for an account.

Key	Value
ident	< Your username >
secret	< Your secret API token >

Method: POST (also: /addDomain)

This is used to dynamically create XirSys domains for an account.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Domain to be created >

Method: DELETE

Deletes a domain for an account (makes inactive).

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Domain to be deleted >

/application

Method: GET (also: /listApplications)

Returns a list of all applications for a domain.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Parent domain >

Method: POST (also: /addApplication)

This is used to dynamically create XirSys applications for a domain.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Parent domain >
application	< Application to be created >

Method: DELETE

Deletes an application for a domain (makes inactive).

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Parent domain >
application	< Application to be deleted >

/room

Method: GET (also: /listRooms)

Returns a list of all rooms for an application.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Top level domain >
application	< Parent application >

Method: POST (also: /addApplication)

This is used to dynamically create XirSys rooms for an application.

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Top level domain >
application	< Parent application >
room	< Room to be created >

Method: DELETE

Deletes a room for an application (makes inactive).

Key	Value
ident	< Your username >
secret	< Your secret API token >
domain	< Top level domain >
application	< Parent application >
room	< Room to be deleted >

Signalling

The XirSys signalling platform is currently the least utilised service of XirSys, in part due to the lack of sufficient documentation. However, over the coming months, the signalling API is most likely to be the service which expands most.

The current API for the signalling platform is very simple, but first, a user must connect with a token, by utilising the /signal/token and /signal/list endpoints detailed above. Once connected, users can expect messages of the format listed below.

Package Structure

Signalling packages are in the JSON format and are structured as:

```
{
   "t" : “u”, // string : type of message
   "o" : “/xirsys.com/default/default/norbert”, // string : address of receiver
   "m" : “peers”, // string | integer | object : meta data
   "k" : “system”, // string : address of sender | “system”
   "i" : “09533210-be7b-11e4-80e4-5f6a3db4a43e”, // string : unique message id
   "d" : { // string | integer | object : packet data
      “users” : [“/xirsys.com/default/default/cecil”]
   }
}
```

Packages may contain further values for XirSys internal use purposes. All incoming messages can be filtered by type and meta. Currently, all signalling messages are of type “u”, meaning “user”. The meta parameter signifies announcements, such as “peers”, “peer_connected” and “peer_removed”. The latter two are for when a peer has connected or disconnected, respectively. When these are sent, the “d” parameter provides an object listing the connected peers name / identifier. The “peers” meta value, however, is sent once, upon first connecting to the signalling, and contains an array of the name / identifiers of all current users in the room.

Custom messages can be used when sending your own messages. Look at the xirsys.signal.js library, provided in the xsdk repository (http://github.com/xirdev/xsdk) to see how this works. The format above is how the XirSys internal system expects messages to be, but there are no restrictions on what data or meta should be, so long as the packet adheres to the above format.

XirSys STUN/TURN Servers

Our Beta STUN/TURN servers have been written from the ground up, which means they are very different from our public servers. We rewrote this system in order to make it simpler and quicker to create new features and connection patterns, which are necessary to make WebRTC more useful.

As these servers are new, they will likely contain bugs and under-perform. The servers are not yet battle hardened or totally optimised. However, that said, they are built soundly according to the STUN and TURN RFC specifications and are very capable systems.

One of the exciting new features we will be adding very soon to the Beta will be the client logging. This means, it will be possible to see what occurs within the STUN/TURN servers when a connection is requested. This may be nothing, should the client fail to even reach the TURN server due to NAT restrictions, or it may be a trove of valuable information which is crucial for debugging client platforms - certainly a big missing piece in today's WebRTC infrastructures.

Publishing - not yet deployed

Publishing is the means to broadcast data from one or more users to many subscribers. This functionality works differently to typical WebRTC connections where each connection between individual users constitutes a physical connection. Therefore, many connections to a specific publisher simply creates too many physical connections for the publishers device to handle.

The XirSys publishing platform works differently. Users wishing to publish first specify the type “publish” (see /signal/token REST call for more details) when requesting a signalling token from the XirSys servers. Once connected to the signalling platform with the token, the user is then able to publish his/her stream by making a standard WebRTC handshake to the user “MCU”. The MCU user is an automated platform which will handshake and receive the stream provided by the publishing user. Offers sent to the MCU user must have a message attribute value of “offer”.

Further users may then subscribe or publish to the MCU in the same manner. While multiple publishers are allowed, only the first publisher will initially be broadcast to subscribers. The designated publisher may then be swapped at any time by sending a packet with the meta value “set_publisher” and the data value as the string identifier of the person to publish. Specifying the identifier of a subscriber or changing the publisher via a subscribers signalling connection will cause the message to be silently discarded.

General Broadcasting

It is estimated that approximately 300 - 500 subscribers may be supported by the above platform. For larger numbers of subscribers, The XirSys platform can provide a general broadcast URL for subscribers to view the published stream using the HTML5 video tag (view /subscribe REST endpoint). The downside to this technology is that the latency from publish to viewing will increase by several seconds as the MCU transcodes the stream into an alternative format.

Roadmap

This page lists several of our upcoming features to be released over the next two to six months. This is not a complete list, but outlines only those features we feel would be of immediate value to our customers.

WebRTC TURN / STUN additions

- DTLS support (already available in our Public service)
- Persisted Recording of streams
- Reliable One to many routing of video/audio (publish & subscribe)
- Robust Many to many muxing of video/audio (MCU features).

REST API

- Per account analytics endpoints for all data transfer
    - Request usage by period
    - Request usage by session
- Per account logging of all internal systems (Signalling, TURN server etc.).

Signalling

- Namespace based message distribution.
    - Message to all in the domain.
    - Message to all in the application.
- SimpleWebRTC SignalMaster API.
- Temporary volatile data storage / metadata by namespace (by domain, application, room, user etc.).
- Persistent data storage by namespace (by domain, application, room, user etc.).
- Alternative message distribution patterns for use with or without WebRTC functionality.

Client support

- Full JavaScript SDK for Signalling and WebRTC
- iOS SDK
- Android SDK

Contact

We welcome all feedback of every kind at XirSys. If you have a query, conundrum, comment or revelation, please feel free to drop us an email at experts@xirsys.com.