XirSys v2 Platform Documentation

Version 1.0.0

Version: 1.0.0
Updated: 20th June, 2015
Author: Lee Sylvester

Contents

Changes

As we progress with the v2 platform, new features will be added. Where changes affect the public platform, we will provide additional legacy hooks to ensure backward compatibility.

All visible changes will be listed below:

v1.0 Saturday, 20th June, 2015

Beta platform launched on public servers.

v0.2b Wednesday, 25th February, 2015

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

v0.2.2b Monday, 30th March, 2015

Signalling: Added pub/sub handshaking.

Introduction

Welcome to the XirSys v2 platform. This new system has been in development for some time and greatly improves on our original offering, providing exception scalability and reliability, as well as a means to be able to rapidly add new features and offerings. We have several such features in development right now, so please do keep an eye open for new blog posts.

At XirSys, we aim to provide a service that developers want, not just for WebRTC, but for RTC in general. If you have any suggestions or requirements that you would like to see us provide, please do let us know at experts@xirsys.com.

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 and POST (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 and POST (also: /wsList)

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

/ice

Method: GET and POST (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)

/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 : system message type
   "m" : { // string | integer | object : meta data
      "t" : "/www.xirsys.com/default/default/norbert", // string : address of receiver
      "s" : null, // timestamp
      "o" : "peers", // local message type
      "i" : null, // string : unique message id
      "f" : "system" // string : address of sender | “system”
   },
   "p" : { // 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 “p” 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 packet data should be, so long as the packet adheres to the above format.

XirSys STUN/TURN Servers

Our new STUN/TURN servers have been written from the ground up, in order to make it simpler and quicker to create new features and connection patterns, which are necessary to make WebRTC more useful.

One of the exciting new features we will be adding very soon 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.

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.