Real-time video chat XPage application using websocket and WebRTC technologies AD-1077

Dr Csaba Kiss 02/03/2016

HTML

WEBSOCKET

WebRTC

Connect 2016

The Premier Social Business and Digital Experience Conference

Los Alamos
NATIONAL LABORATORY
EST. 1943

Make
Every
Moment
Count

#ibmconnect

LA-UR-16-20047

Credentials

- Over 25 years experience in molecular biology
- Began Xpage application development in 2014
- Self-taught JavaScript enthusiast
- Twitter: @csakis
- Blog: XpageXplorer.org

Make Every **Moment** Count

Websocket survey

~~websocket-survey.herokuapp.com~~

ws-survey.mybluemix.net

Make Every **Moment** Count

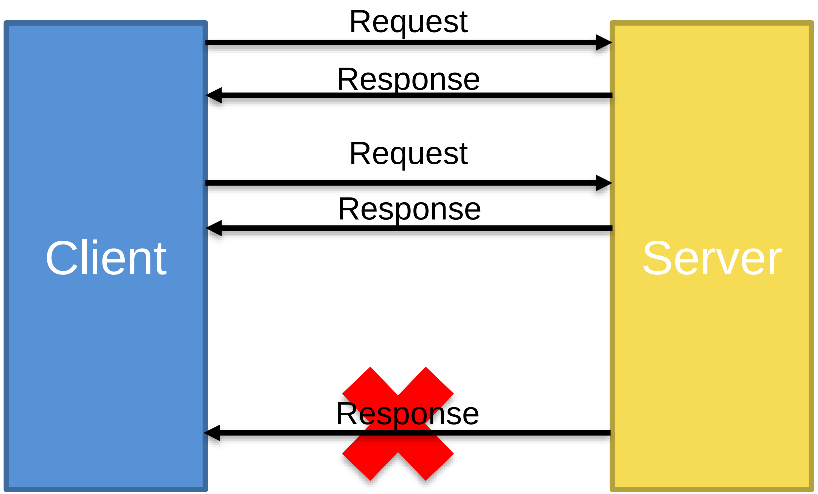
Agenda

- HTTP protocol drawbacks
- Websocket
    - overview
    - API
    - Installing OpenNTF plugin
    - Websocket code examples
    - Serverside listeners using SSJS
    - Pros and cons
- WebRTC
- DEMO

Make Every **Moment** Count

HTTP protocol

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP functions as a request-response protocol in the client-server computing model.*



*: wikipedia

Too much overhead

```
× Headers Preview Response Cookies Timing
▼General
    Request URL: http://csaba-pc/WebRTCapp.nsf/test.xsp
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: 192.168.0.2:80
▼Response Headers     view parsed
    HTTP/1.1 200 OK
    Server: Lotus-Domino
    Date: Mon, 04 Jan 2016 18:21:31 GMT
    Content-Type: text/html;charset=UTF-8
    Expires: -1
    Content-Encoding: gzip
    Content-Length: 596
▼Request Headers     view parsed
    GET /WebRTCapp.nsf/test.xsp HTTP/1.1
    Host: csaba-pc
    Connection: keep-alive
    Cache-Control: max-age=0
    Authorization: Basic Q3NhYmEgS2lzczpFdGFxcTIzNA==
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
    Upgrade-Insecure-Requests: 1
    User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36
    DNT: 1
    Accept-Encoding: gzip, deflate, sdch
    Accept-Language: en-US,en;q=0.8,hu;q=0.6,sv;q=0.4
    Cookie: SessionID=75BFB4F2DAF12D0746B103B96247890F1136D210
```

Clients	Req/min*	MB/min
100	600	5
500	30,000	26
1,000	60,000	52
10,000	600,000	522

871 bytes header data (without any cookie)

*: 1 request every second

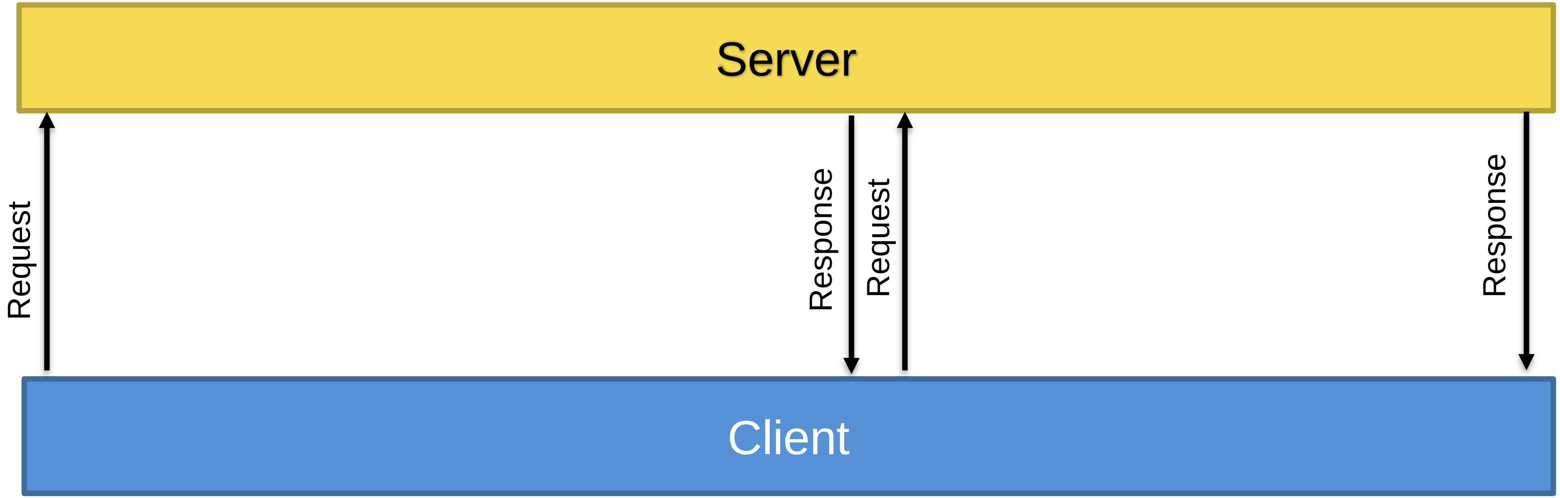
Other HTTP limitations

- Every request needs a new connection (latency)
- Half duplex connection (walkie talkie)

Make Every **Moment** Count

Work arounds

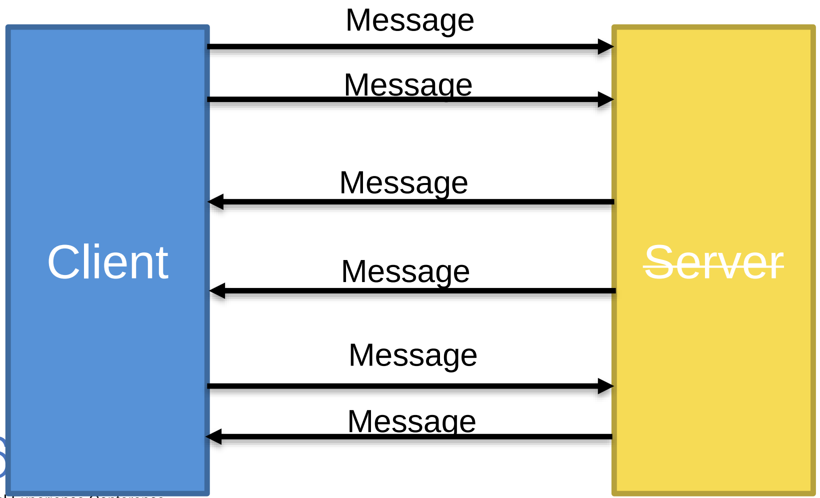
Long-polling (comet)



Complicated implementation, Not standardized.

General websocket

- WebSocket is a protocol providing **full-duplex** communication channels over a **single** TCP connection.
- Both the WebSocket API itself (W3C) and the WebSocket protocol are **standards**, see RFC 6455.
- The WebSocket protocol is currently supported in most major browsers



Browser compatibility*

Web Sockets - CR

Bidirectional communication technology for web apps

U.S.A.	89.42%	+	2.59%	=	92.01%
unprefixed:	89.42%	+	2.54%	=	91.96%
Global	87.72%	+	0.94%	=	88.66%
unprefixed:	87.72%	+	0.86%	=	88.58%

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			40						
8			41						
9			45	[2] 6.1		7.1			
10		42	46	8		8.4		4.4	
11	13	43	47	9	34	9.2	8	46	47
	14	44	48		35				
		45	49		36				
		46	50						

*:caniuse.com

Websocket API

Connect 2016

The Premier Social Business and Digital Experience Conference

Make
Every
Moment
Count

#ibmconnect

Websocket constructor

Establishing a new websocket connection

```
var ws = new WebSocket("ws://localhost:8080", ['soap', 'myWsProtocol']);
```

1 2 3 4

1. ws is the new websocket object
2. ws:// denotes websocket protocol
3. websocket port
4. optional protocols

The hansdshake

- The client sends a WebSocket handshake request:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

- The server responds:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

- Connection is “upgraded”

Websocket is purely event driven

4 events

```
ws.onopen = function () {
  //body
};

ws.onmessage = function(msg) {
  //do something
};

ws.onerror = function (err) {
  // log error
};

ws.onclose = function () {
  //body
}
```

Make Every **Moment** Count

Websocket methods

2 methods

```
ws.send(message);

ws.close([code]);
```

The websocket message format is important with the OpenNTF websocket plugin.

Optional close codes:

- 1000 – CLOSE_NORMAL
- 1006 – CLOSE_ABNORMAL
- 1012 – SERVICE_RESTART
- …

Simple websocket message format

```
{
  "from": "CN=Csaba Kiss/O=CSABA-PC",
  "to": "broadcast",
  "text": "test message",
  "date": 1451945109238
}
```

- Message in JSON format
- Sender format needs to be in canonical format @UserName()
- Recipient:
  - broadcast (everybody receives it)
  - Canonical user name (specific user)
  - Role based messaging (see websocket-setup.pdf)
- Date: omitted || Unix epoch || yyyy-MM-dd hh:mm a

Complex websocket message

- Embedded data object
- Binary data transfer
- Sending other attributes
  - Application
  - Message type

  …


```json
{
  "to": "broadcast",
  "from": "CN=Csaba Kiss/O=CSABA-PC",
  "date": 1451948246736,
  "text": "",
  "data": {
    "application": "webrtcapp",
    "type": "status",
    "state": "cameraOn"
  }
}
```


- Websocket server is not application specific!!

Make Every **Moment** Count

Websocket attributes

2 attributes

`readyState`

- CONNECTING
- OPEN
- CLOSING
- CLOSED

`bufferedAmount`

returns the number of bytes that have been queued but not yet sent.

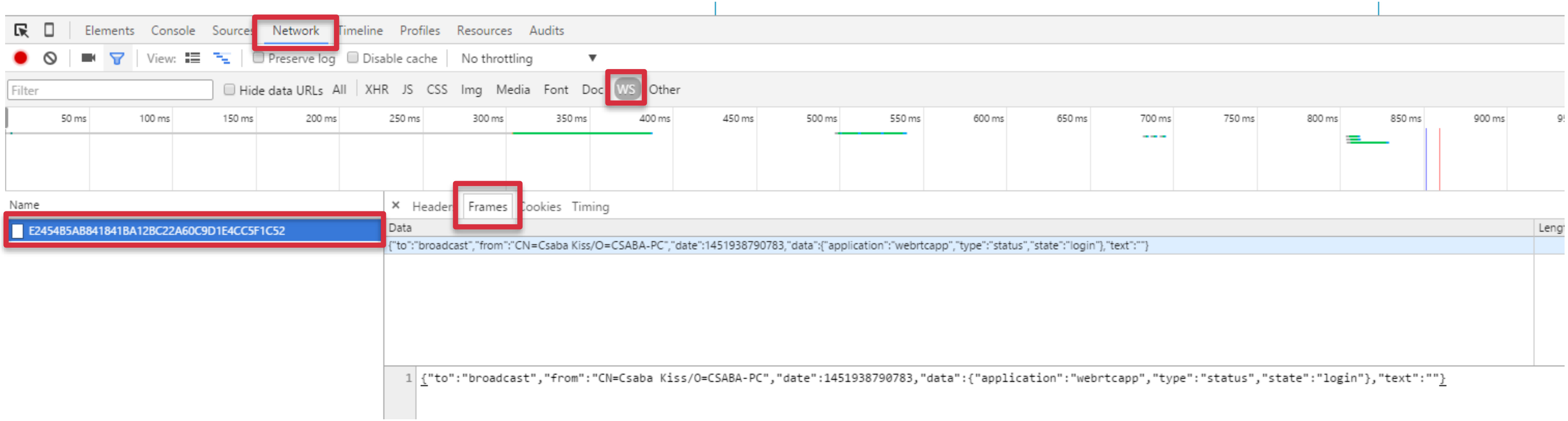
Useful to prevent network saturation

```
if (ws.readyState === "OPEN") {
  ws.send(socketMessage);
}
```

```
var THRESHOLD = 10000;
if (ws.bufferedAmount < THRESHOLD) {
  ws.send(socketMessage);
}
```

Websocket debugging

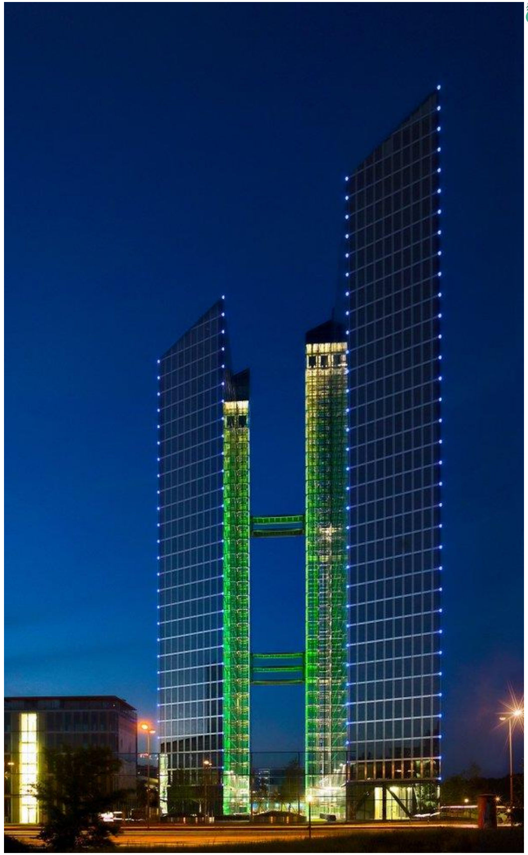
Use Chrome!!



Make Every **Moment** Count

Potential applications

- Real time communication
- Trading
- Auction sites
- Gaming
- Collaborations
- IoT (Internet of Things)



Watson IoT HQ in Munich

Potential problems and pitfalls

- Proxies and firewalls:
  - Long-lived connections might not be allowed
  - Designed for HTTP traffic
  - Might filter out other traffic

- Tip:
  - Use wss://

Websocket servers plugin

Connect 2016

The Premier Social Business and Digital Experience Conference

Make Every **Moment** Count

#ibmconnect

OpenNTF websocket extension library*

https://www.openntf.org/main.nsf/project.xsp?r=project/webshell-xpages-ext-lib

GitHub:

https://github.com/mwambler/webshell-xpages-ext-lib

Domino Implementation of Java-Websocket server http://java-websocket.org

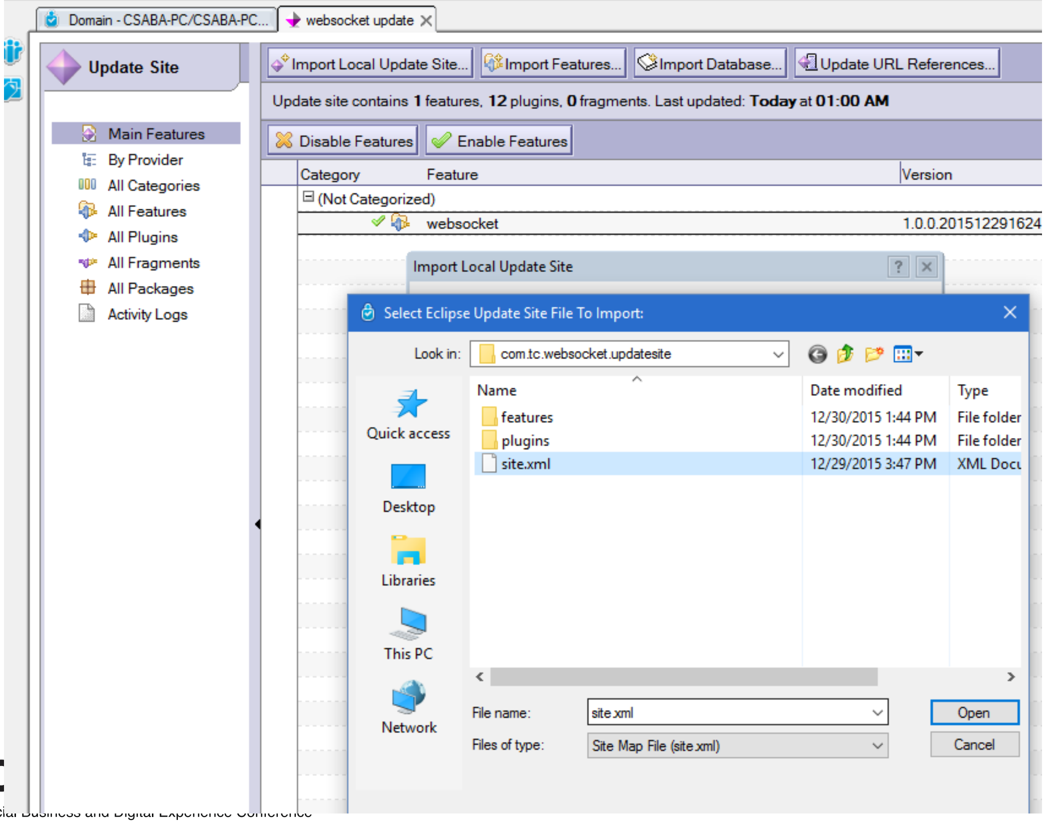
Support

Mark Ambler
Tek Counsel LLC
Twitter: @mwambler
Blog: http://markwambler.blogspot.com
Email:mambler@tekcounsel.net

*: IBM Domino server does not support websocket protocol

Plugin installation on Domino server part 1

Create Eclipse update site: websockupdate.nsf



Plugin installation on Domino server part 2

Install OpenNTF extension library

Copy websocket.ntf file from `xpage-applications` folder to server data folder

Modify `notes.ini`:

```
XPagesPreload=1
XPagesPreloadDB=websocket.nsf
OSGI_HTTP_DYNAMIC_BUNDLES=extlibupdate.nsf, websockupdate.nsf
```

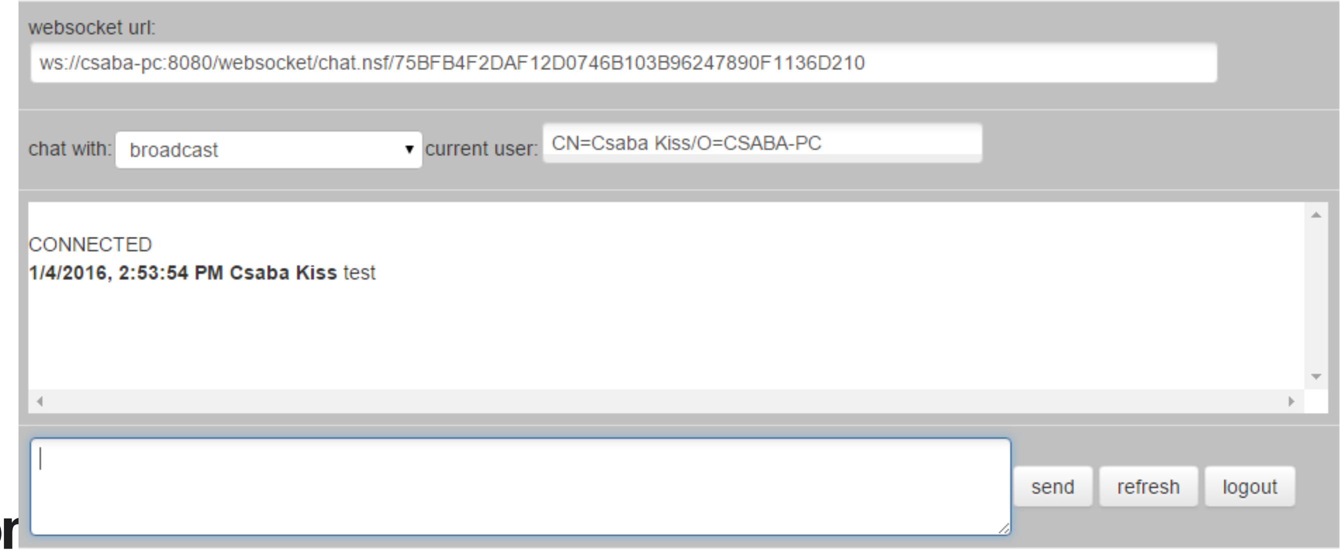
Modify java.policy:

```
grant {
 permission java.security.AllPermission;
};
```

Make Every **Moment** Count

Test Domino websocket extension

- Set appropriate ACLs
- Restart server
- Use included chat.nsf application to test if websocket connection can be established



Make Every **Moment** Count

Websocket settings

Defaults:

```
WEBSOCKET_PORT=8889
WEBSOCKET_MAX_CONNECTIONS=100
WEBSOCKET_MAX_MSG_SIZE=1048576
WEBSOCKET_ENCRYPT=false
WEBSOCKET_ALLOW_ANONYMOUS=false
WEBSOCKET_CLUSTERED=false
WEBSOCKET_FILTER=null
```

Optional:

- secure connection
- cluster support

Make Every **Moment** Count

Console command line operations

tell http osgi websocket stop/start

tell http osgi websocket count/count-all

tell http osgi websocket show-users/show-all-users

tell http osgi websocket register-script localhost /path/app.nsf */path/app.nsf/ssjs

tell http osgi websocket reload-scripts

tell http osgi websocket remove-script localhost /path/app.nsf */path/app.nsf/ssjs

Persistence using server side listener

- Communicate directly to the Domino database(!!)
- Server side listener uses Rhino JavaScript Engine
    - No access to scope variables or @functions
    - Cannot define variables with : notation:

      (i.e. var doc:NotesDocument = currentDocument.getDocument();)

- REST API (/api/websocket/v1/sendmessage)
- Targeted messaging by URI (filter by roles, page)
- Initialization:

```
if (!websocketBean.containsSocketListener("/WebRTCapp.nsf/applicationSSJS")) {
  websocketBean.addSocketEventListener("/WebRTCapp.nsf", "*", "/WebRTCapp.nsf/applicationSSJS");
}
```

Make Every **Moment** Count

ServerSide gotchas

Check that the message is coming from the appropriate application.

Received message = `socketMessage`

Sender = `socketMessage.getFrom();`

chatMessage = `socketMessage.getText();`

Getting Data attribute example:

```
var application = socketMessage.getData().get("application");
```

Xpage websocket demo

Connect 2016

The Premier Social Business and Digital Experience Conference

Make Every **Moment** Count

#ibmconnect

Websocket code example

Connect 2016

The Premier Social Business and Digital Experience Conference

Make Every **Moment** Count

#ibmconnect

User tracking with persistence on Domino Server

User form

Name (String)

Online (yes/no)

Idle (yes/no)

Camera (yes/no)

InCall (yes/no)

Users view

WebRTCapp
- Chats
- Users

Name	Online	Idle	Camera	InCall
CN=Csaba Kiss/O=csaba	yes	yes	yes	no
CN=Robert Kiss/O=csaba	yes	yes	no	no

User tracking with websocket 2

Client side

- User logs in to application
- Application establishes websocket connection

```
var ws = new Websocket(uri);
```

- Websocket onOpen event fires. Client sends out status message

```
ws.onOpen : function() {
    ws.send(createStatusMessage("login"));
  }
```

User tracking with websocket 3

Server Side

receives websocket status message and fires `onMessage` event

```
function onMessage(){
  var application = socketMessage.getData().get("application");
  if (application != "webrtcapp")
    return false;
  var msgType = socketMessage.getData().get("type");
    if (msgType == "status") {
      var msgStatus = socketMessage.getData().get("state");
      if (msgStatus =="login") {
        var db = session.getDatabase("","WebRTCapp.nsf");
        var currentUser = socketMessage.getFrom();
        var userView = db.getView("Users");
        var doc = userView.getDocumentByKey(currentUser);
        doc.replaceItemValue("Online", "yes");
        doc.replaceItemValue("Idle", "no");
        doc.replaceItemValue("Camera", "no");
        doc.save();
      }
```

User tracking with websocket 4

Server Side

Server broadcast statusUpdate message to all clients

```
var srvMsg = websocketClient.createMessage();
srvMsg.setTo("broadcast");
srvMsg.setText("response from server");
srvMsg.getData().put("application","webrtcapp");
srvMsg.getData().put("type","statusMsgFromServer");
websocketClient.sendMsg(serverMessage);
```

User tracking with websocket 5

Client side

- fires `onMessage` event

```
// message type = statusMsgFromServer
if (msg.data.type === "statusMsgFromServer") {
  displayUsers(msg.data.status);
}
```

- Refreshes users list using REST service

```
function displayUsers(response) {
$.getJSON(
    "Home.xsp/getUsers", //refresh userlist with REST
    function(data) {
      ...
    }
```

Make Every **Moment** Count

Websocket conclusion

- Allows you combine http protocol with websocket data traffic.
- Lets developers write event-driven real-time applications.
- Helps you writing single page Xpage applications with no partial refreshes.
- User experience is more fluid and satisfying.
- The OpenNTF websocket extension library allows developers to write native XPage applications with seamless server side Domino database integration.

WebRTC

Connect 2016

The Premier Social Business and Digital Experience Conference

Make
Every
Moment
Count

#ibmconnect

WebRTC definition

WebRTC (Web Real-Time Communication) is an API definition drafted by the World Wide Web Consortium (W3C) that supports **peer-to-peer** applications for voice calling, video chat, and P2P file sharing **without** the need of either internal or external **plugins**.*

“WebRTC is the biggest inflection point that has ever happened for the web platform.”

Kyle Simpson

*:Wikipedia

The good bits

- Open source technology supported by Google, Mozilla, Apple, Cisco, Opera.
- Simple Javascript API built into the browser
- No plugins required.
- Can stream audio/video, dedicated data channel, easy screen sharing
- Platform agnostic

The bad bits

- API and protocol have not been standardized yet.
- Browser implementation is patchy
- Doesn’t scale well.

Browser support

WebRTC Peer-to-peer connections - WD

Method of allowing two users to communicate directly, browser to browser using the RTCPeerConnection API.

U.S.A.	47.96%
unprefixed:	0%
Global	57.01%
unprefixed:	0%

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			40						
8			41						
9			45	6.1		7.1			
10		42	46	8		8.4		4.4	
11	13	43	47	9	34	9.2	8	46	47
	14	44	48		35				
		45	49		36				
		46	50						

Latest from Microsoft

Microsoft | Developer

Search

IEBlog

Internet Explorer Team Blog

Bringing Interoperable Real-Time Communications to the Web

Rate this article ★★★★★

October 27, 2014 By ieblog

f 0 | 0 | in 0 | 32

Together with the industry-leading expertise of Skype, we're excited to announce development has begun on the ORTC API for WebRTC, a key technology to make Real-Time Communications (RTC) on the web a reality.

We aim to make browser-based calls more convenient by removing the need to download a plugin. It's all about convenience – imagine you'll be able to simply open IE and make a Skype call to friends, family, or get real-time support for that new device right from your browser.

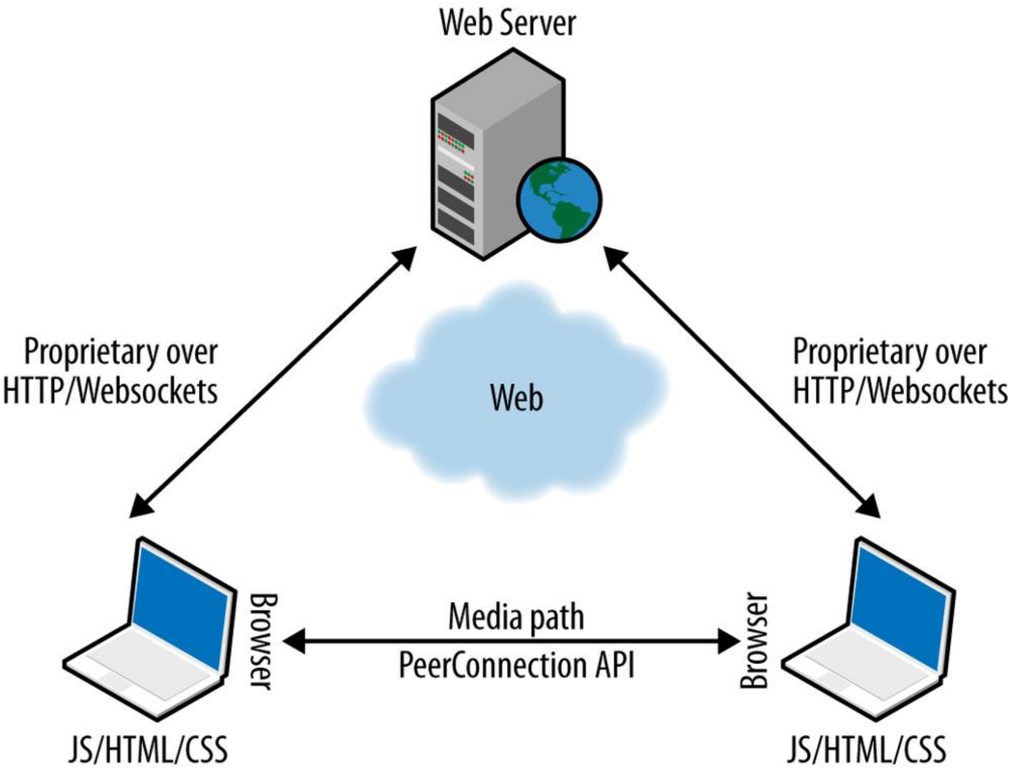
ORTC API for WebRTC

We've been actively collaborating with the W3C and IETF to contribute and improve standards like the ORTC API for WebRTC to enable a wide range of features from simple conversations to scalable multiparty video conferences. With the momentum from over 80 participants that represent a variety of browsers, communications experts and start-ups, the W3C ORTC Community Group has issued a "Call for Implementations," which signals the ORTC specification has reached significant stability. Building on top of the experiences from Skype and Lync and prototyping effort done by the Microsoft OpenTech, we are now working to deliver the ORTC API in Internet Explorer.

The ORTC specification supports the underlying protocols as defined by the IETF RTCWEB Working Group, which enables support for advanced video conferencing technologies such as simulcast and scalable video coding. We are working with the web community on various fronts to influence how a subset of ORTC objects and methods become part of the WebRTC 1.0 API. This helps to provide a seamless transition from WebRTC 1.0 to a JavaScript object-based real-time communications model based on ORTC (i.e. WebRTC 1.1).

Make Every **Moment** Count

The WebRTC triangle



Salvatore Loreto; Simon Pietro Romano:Real-Time Communication with WebRTC

Video chat demonstration

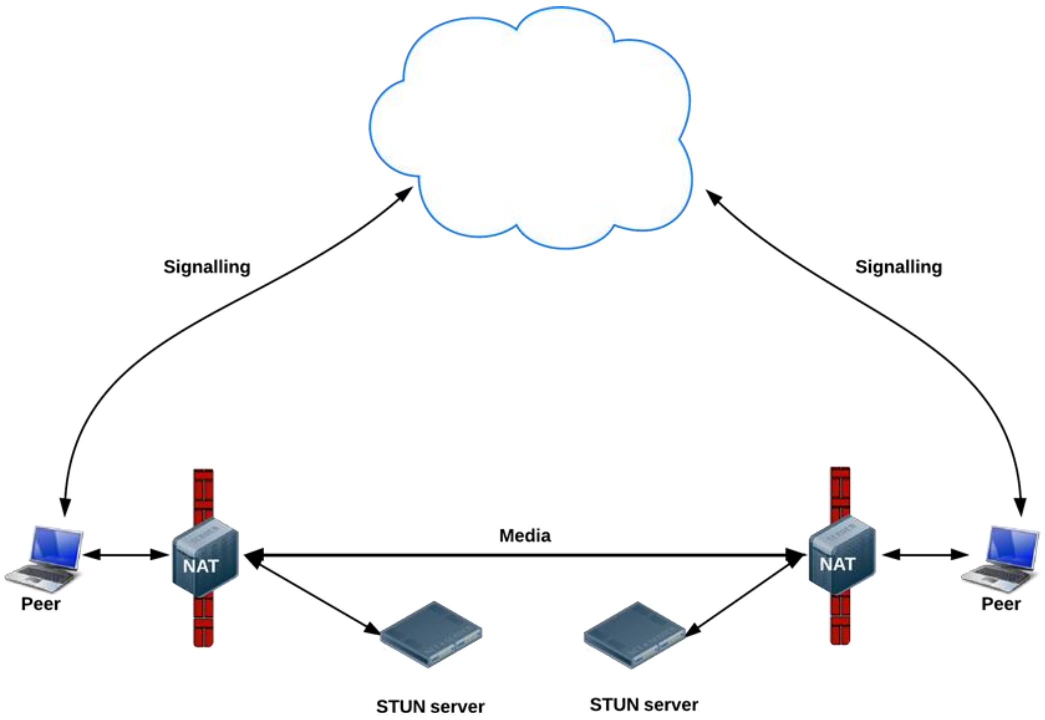
Connect 2016

The Premier Social Business and Digital Experience Conference

Make Every **Moment** Count

#ibmconnect

Connection via STUN* server



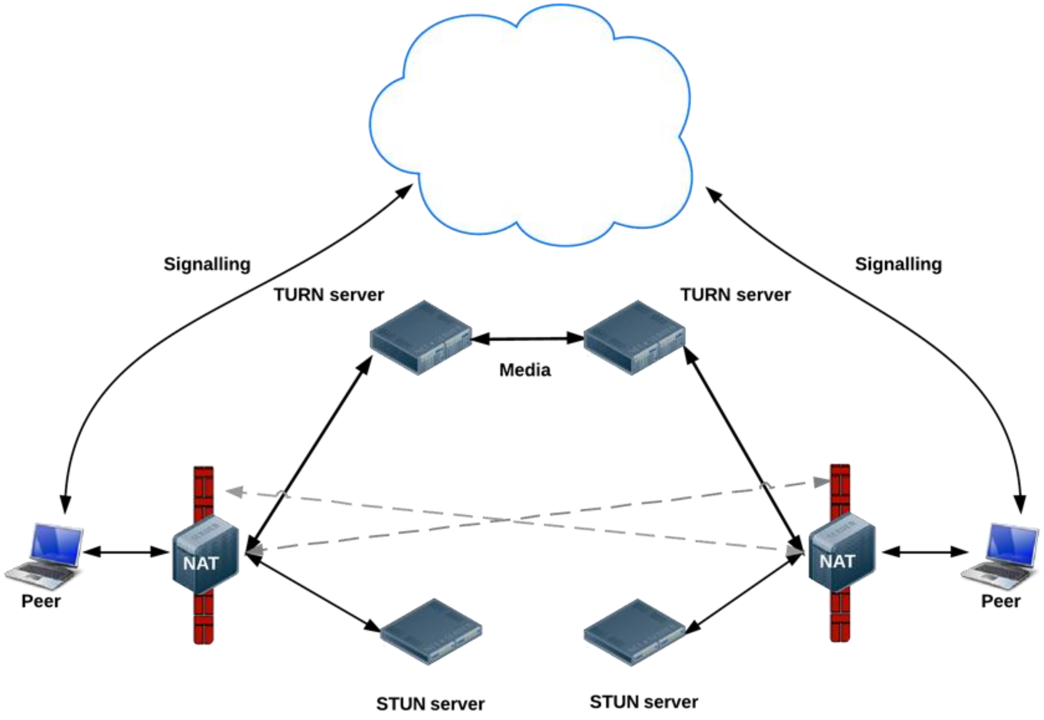
- Valid peer-to-peer connection
- Public stun servers available
- Build you own STUN server

https://github.com/coturn/coturn

Make Every **Moment** Count

STUN: Session Traversal Utilities for NAT

Connection using a TURN* server



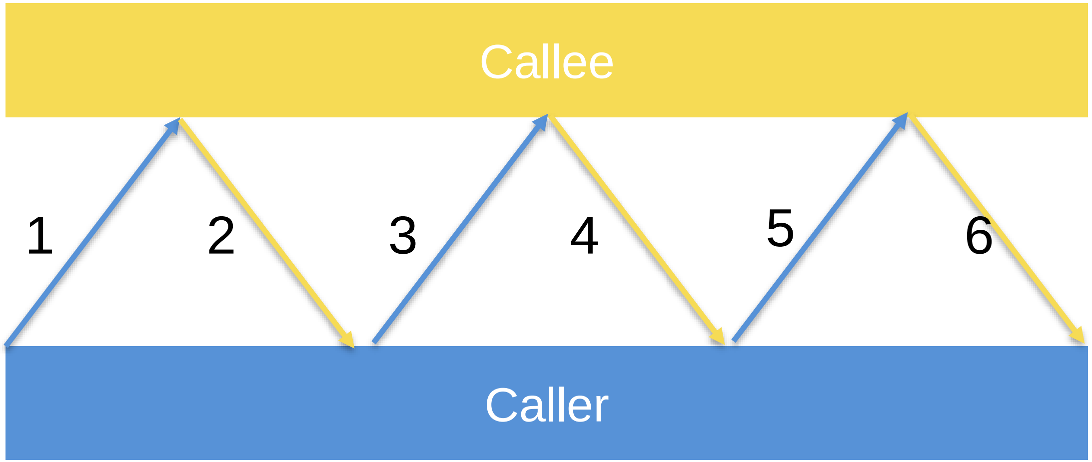
- Video streams through server
- High bandwidth usage
- Fallback if STUN does not work

Make Every **Moment** Count

TURN: Traversal Using Relays around NAT

Signaling protocol

- BYOS: Bring your own signaling
- Signaling is not part of the WebRTC standard
- Websocket API is perfect for writing signaling cascade, since it’s based on events.



1. Caller places call
2. Callee answers call
3. Caller send offer
4. Callee send answer
5. …
6. …

WebRTC conclusion

- WebRTC is an emerging technology.
- It is not ready for prime time.
- If all your customers use Chrome/Firefox, then you can give it a try.
- The websocket server extension is ideal for establishing WebRTC connection.

Thank you for your attention!

Please remember to fill out your feedback form

www.connectsurveys.com

Questions/Comments

Contact

Twitter: @csakis

Blog: http://XpageXplorer.org