rtcweb
Internet-Draft
Intended status: Standards Track
Expires: August 2, 2013

K. Li
Huawei Technologies

January 29, 2013

RTCWeb JSEP XMPP/Jingle Mapping
draft-li-rtcweb-jsep-xmpp-mapping-02

Abstract

This document proposes mapping message representations between RTCWeb Javascript Session Establishment Protocol(JSEP) scheme and XMPP/ Jingle [XEP-0166] messaging scheme. Such a signaling mapping is intended to enable Javascript to use XMPP/Jingle to establish a session between two RTCWeb enabled browsers.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 2, 2013.

Table of Contents

1. Introduction

In draft [I-D.ietf-rtcweb-jsep], it is mentioned that there are several options for the signalling mechanisms: ROAP (see [I-D.jennings-rtcweb-signaling]), SIP or XMPP/Jingle.

This document focuses on XMPP/Jingle and tries to explain how to use JSEP and XMPP/Jingle to exchange session descriptions.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Architecture Overview

2.1. Architecture Model

In Figure 1, it shows the overall architecture. In the figure, "Browser" is synonymous with "User Agent", and "Web Application" is synonymous with "JavaScript".

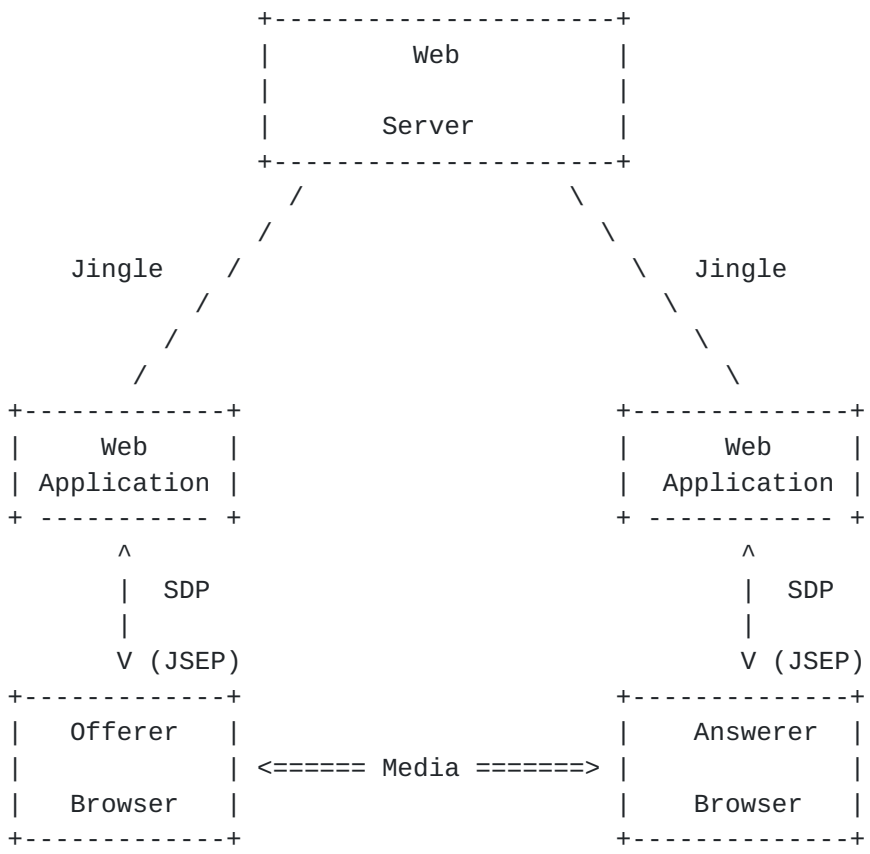

Figure 1: JSEP-XMPP/Jingle Mapping Architecture

2.2. Basic Session Flow

In Figure 2, it shows the basic Jingle session flow.

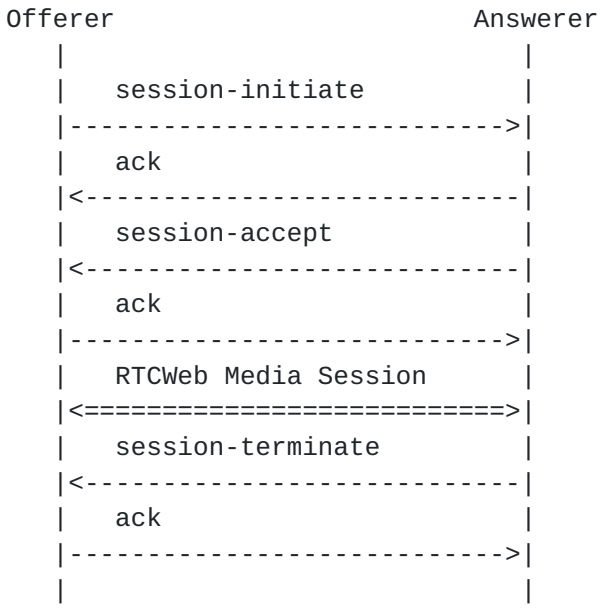

Figure 2: Jingle Session Flow

2.3. Overall Session Management

In Figure 3, it shows the overall Jingle session management.

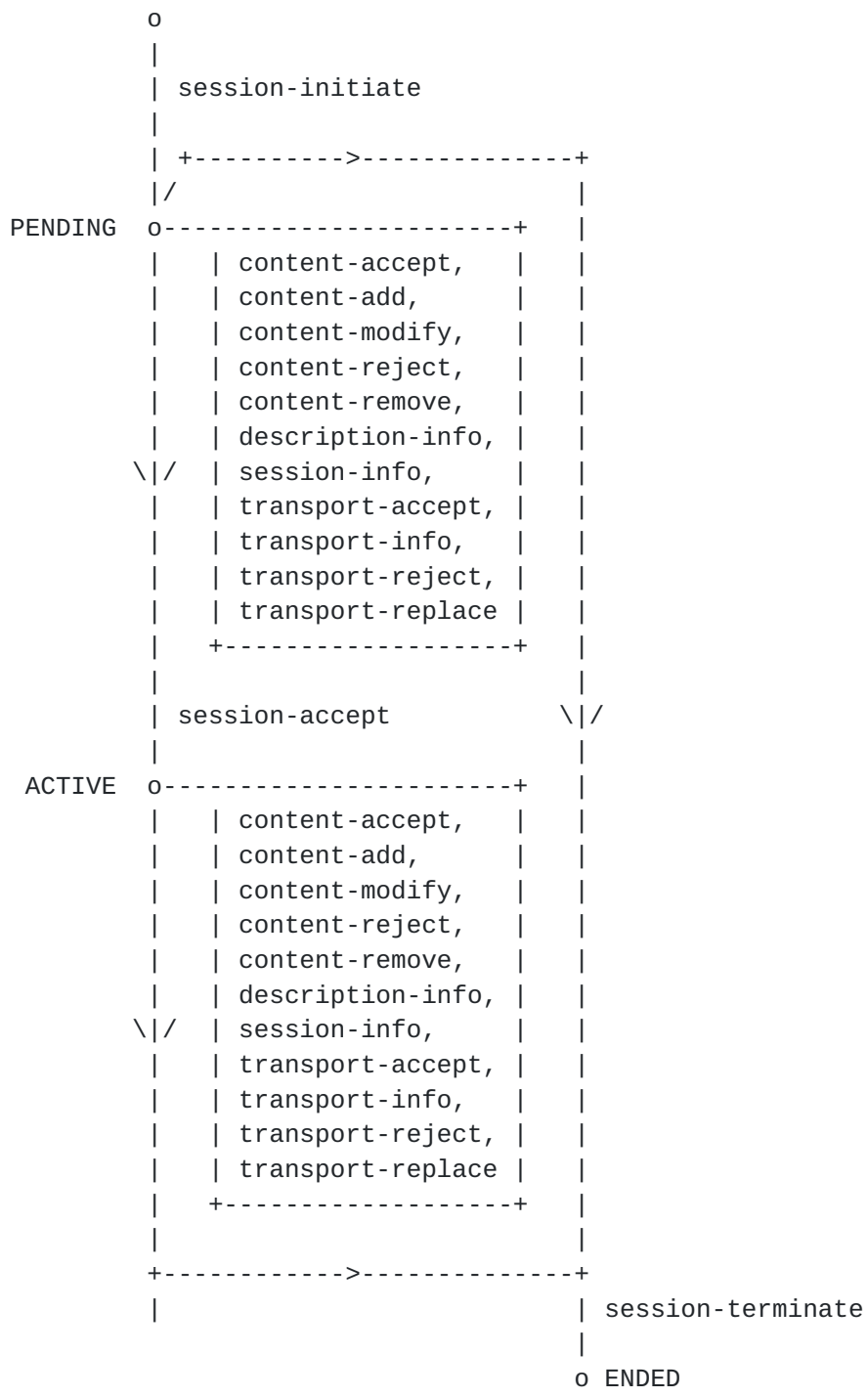

Figure 3: Jingle Overall Session Management

In Section 3, it introduces how JS clients could use the Jingle actions to manage a session. The detailed descriptions of the Jingle actions are defined in [XEP-0166].

3. Media Setup

3.1. Session Management

3.1.1. Initiate the Session

To initiate a session, the initiator can create an offer, and send the offer to the recipient by using Jingle "session-initiate" action.

The JSEP APIs are defined in [webrtc-api] and [I-D.ietf-rtcweb-jsep].

JSEP API:

OffererJS->OffererUA: pc = new PeerConnection();

OffererJS->OffererUA: pc.addStream(localStream, null);

OffererJS->OffererUA: offer = pc.createOffer(null);

Jingle message:

OffererJS->AnswererJS: <jingle action="session-initiate"/> .

After receiving the Jingle "session-initiate" action, the recipient can parse the session information, and apply the supplied offer as the remote description.

JSEP API:

AnswererJS->AnswererUA: pc.setRemoteDescription("offer",offer);

3.1.2. Accept the Session

If the recipient accepts a session, it can create an answer and send back the answer by using Jingle "session-accept" action.

JSEP API:

AnswererJS->AnswererUA: peer.addStream(localStream, null);

AnswererJS->AnswererUA: answer = peer.createAnswer(offer, null);

Jingle message:

AnswererJS->OffererJS: <jingle action="session-accept"/>.

After receiving the Jingle "session-accept" action, the initiator can parse the received answer and apply the supplied answer to the remote

description.

JSEP API:

OffererJS->OffererUA: pc.setRemoteDescription("answer", answer);

3.1.3. Terminate the Session

To terminate a session, the initiator can close the peer connection with the recipient by using Jingle "session-terminate" action.

JSEP API:

OffererJS->OffererUA: pc.close();

Jingle message:

OffererJS->AnswererJS: <jingle action="session-terminate"/>.

After receiving the Jingle "session-terminate" action, the recipient can close the peer connection.

JSEP API:

AnswererJS->AnswererUA: peer.close();

3.2. Media Management

3.2.1. Add Media

To add media (e.g.video) to an existing session, the initiator can use Jingle "content-add" action.

JSEP API:

OffererJS->OffererUA: pc.addStream(videoStream);

OffererJS->OffererUA: offer = pc.createOffer(null);

Jingle message:

OffererJS->AnswererJS: <jingle action="content-add"/>.

After receiving the Jingle "content-add" action, the recipient can parse the received offer and set the remote description.

JSEP API:

AnswererJS->AnswererUA: peer.setRemoteDescription("offer", offer);

3.2.2. Modify Media

To modify media (e.g.change audio to video) to an existing session, the initiator can either use Jingle "content-modify" action, or use combined Jingle "content-remove" action and "content-add" action.

JSEP API:

OffererJS->OffererUA: pc.removeStream(audioStream);

OffererJS->OffererUA: pc.addStream(videoStream);

Jingle message:

OffererJS->AnswererJS: <jingle action="content-modify"/>.

After receiving the Jingle "content-modify" action, the recipient can parse the received offer and set the remote description.

JSEP API:

AnswererJS->AnswererUA: peer.setRemoteDescription("offer", offer);

3.2.3. Remove Media

To remove media (e.g.video) to an existing session, the initiator can use Jingle "content-remove" action.

JSEP API:

OffererJS->OffererUA: pc.removeStream(audioStream);

Jingle message:

OffererJS->AnswererJS: <jingle action="content-remove"/>.

After receiving the Jingle "content-remove" action, the recipient can parse the received offer and set the remote description.

JSEP API:

AnswererJS->AnswererUA: peer.setRemoteDescription("offer", offer);

3.2.4. Accept Media

If the recipient accepts the "content-add" action to an existing session from the initiator, recipient can create an answer and send back the answer by using Jingle "content-accept" action.

JSEP API:

AnswererJS: offer = parseContentAdd(xmpp);

AnswererJS->AnswererUA: peer.createAnswer(offer,null);

Jingle message:

AnswererJS->OffererJS: <jingle action="content-accept"/>.

After receiving the Jingle "content-accept" action, the initiator can parse the received answer and apply the received answer to the remote description.

JSEP API:

OffererJS->OffererUA: peer.setRemoteDescription("answer", answer);

3.2.5. Reject Media

If the recipient rejects the "content-add" action to an existing session from the initiator, recipient can send back answer by using Jingle "content-reject" action.

JSEP API:

AnswererJS: offer = parseContentAdd(xmpp);

AnswererJS->AnswererUA: peer.createAnswer(offer,null);

Jingle message:

AnswererJS->OffererJS: <jingle action="content-reject"/>.

After receiving the Jingle "content-reject" action, the initiator can parse the received answer and apply the received answer to the remote description.

JSEP API:

OffererJS->OffererUA: peer.setRemoteDescription("answer", answer);

3.3. Information Exchange

3.3.1. Exchange the ICE

To perform the ICE process, the initiator can start gathering or update ICE address, and exchange the ICE candidates with the recipient by using Jingle "transport-info" action.

JSEP API:

OffererJS->OffererUA: pc.startIce();

OffererJS->OffererUA: pc.updateIce();

Jingle message:

OffererJS->AnswererJS: <jingle action="transport-info"/>.

AnswererJS->OffererJS: <jingle action="transport-info"/>.

After receiving the Jingle "transport-info" action, the recipient can parse the received ICE candidates and add remote candidate to the ICE Agent.

JSEP API:

AnswererJS->AnswererUA: pc.addIceCandidate(candidate);

3.3.2. Description Information

To send informational hints about parameters related to an existing session, for example, add new video sources to a call that already has video, the initiator can indicate that by using Jingle "description-info" action.

JSEP API:

OffererJS->OffererUA: pc.addStream(offererVideoStream2);

OffererJS->OffererUA: offer = pc.createOffer(null);

Jingle message:

OffererJS->AnswererJS: <jingle action="description-info"/>.

After receiving the Jingle "description-info" action, the recipient parses the description information and sends back the acknowledgement by using IQ stanza of "result" type. There is no mapped JSEP API for

Jingle "description-info" action.

3.3.3. Result

To acknowledge the description information to an existing session from the initiator, recipient can send back answer by using IQ stanza of "result" type. See [RFC6120].

Jingle message:

AnswererJS->OffererJS: <iq type="result"/>.

After receiving the Jingle IQ stanza of "result" type, the recipient can use the remote offer as an answer in the remote description.

JSEP API:

AnswererJS->AnswererUA: peer.setRemoteDescription("answer", offer);

3.3.4. Error

If there are errors occurred during an existing session, the recipient can send back answer by using IQ stanza of "error" type. See [RFC6120].

JSEP API:

AnswererJS->AnswererUA: peer.RTCPeerConnectionErrorCallback;

Jingle message:

AnswererJS->OffererJS: <iq type="error"/>.

After receiving the Jingle IQ stanza of "error" type, the initiate can choose to close the peer connection due to the errors.

JSEP API:

OffererJS->OffererUA: pc.close();

3.4. Other Actions

TBD 1: do we have usage for the following actions: "security-info", "session-info"?

TBD 2: do we need to redefine a transport method? If yes, we can use "transport-replace", "transport-accept", "transport-reject".

4. Mapping between Jingle Message and JSEP API

4.1. Map JSEP API to Jingle Message

When Offerer Javascript uses JSEP API to interact with Offerer User Agent, it needs to map the JSEP API to Jingle message, to send it Answerer JavaScript. In Figure 4, it shows the mapping table from JSEP APIs to Jingle messages.

JSEP API	Jingle Message
createOffer()	session-initiate
startIce()	transport-info
updateIce()	transport-info
createAnswer()	session-accept
close()	session-terminate
addStream()	content-add
removeStream(), addStream()	content-modify
removeStream()	content-remove
createAnswer()	content-accept
createOffer()	description-info
RTCPeerConnectionErrorCallback	iq "error"

Figure 4: Map JSEP API to Jingle Message

4.2. Map Jingle Message to JSEP API

When Answerer Javascript receives Jingle message, it needs to map it to JSEP API, and interacts with ANswerer User Agent. In Figure 5, it shows the mapping table from JSEP APIs to Jingle messages.

Jingle Message	JSEP API
session-initiate	setRemoteDescription()
transport-info	addIceCandidate()
session-accept	setRemoteDescription()
session-terminate	close()
content-add	setRemoteDescription()
content-modify	setRemoteDescription()
content-remove	setRemoteDescription()
content-accept	setRemoteDescription()
description-info	setRemoteDescription()
iq "error"	close()

Figure 5: Map Jingle Message to JSEP API

5. Mapping to SDP

In order to perform the media negotiation, PeerConnection SDP Messages need to be converted into Jingle message and vice-versa.

The session description information included in Jingle message can be mapped to SDP as defined in section 6 of [XEP-0167].

For example, consider a payload of 16-bit linear-encoded stereo audio sampled at 16KHz associated with dynamic payload-type 96:

<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
<payload-type id='96' name='speex' clockrate='16000'/> </description>

That Jingle-formatted information would be mapped to SDP as follows:

m=audio 9999 RTP/AVP 96a=rtpmap:96 speex/16000

6. Example Message Flows

6.1. Exchange Candidates

In Figure 6, OffererJS uses Jingle "session-initiate" action to initiate a session with AnswererJS, and uses Jingle "transport-info" to exchange ICE candidates with AnswererJS. Then AnswererJS accepts the session using Jingle "session-accept" action. After the media session, OffererJS uses "session-terminate" action to terminate the session, and AnswererJS acknowledges with IQ stanza of "result" type.

Figure 6: Exchange Candidates

Message details go here...

6.2. Add Contents

In Figure 7, OffererJS uses Jingle "content-add" action to add video media to an existing session. AnswererJS accepts that by using Jingle "content-accept" action. For simplicity, candidate exchange is not shown.

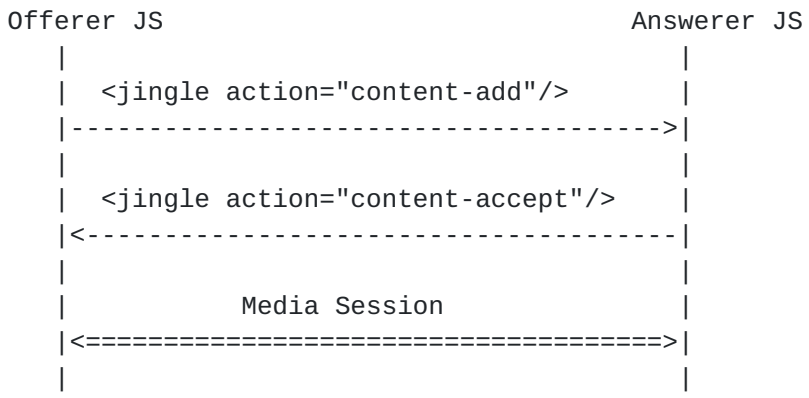

Figure 7: Add Contents

Message details go here...

6.3. Exchange Description Information

In Figure 8, OffererJS uses Jingle "description-info" action to add new video sources at the same time to a call that already has video. AnswererJS also uses Jingle "description-info" action to indicate the new sources to the remote side. After that, they uses IQ stanza of "result" type to acknowledge each other.

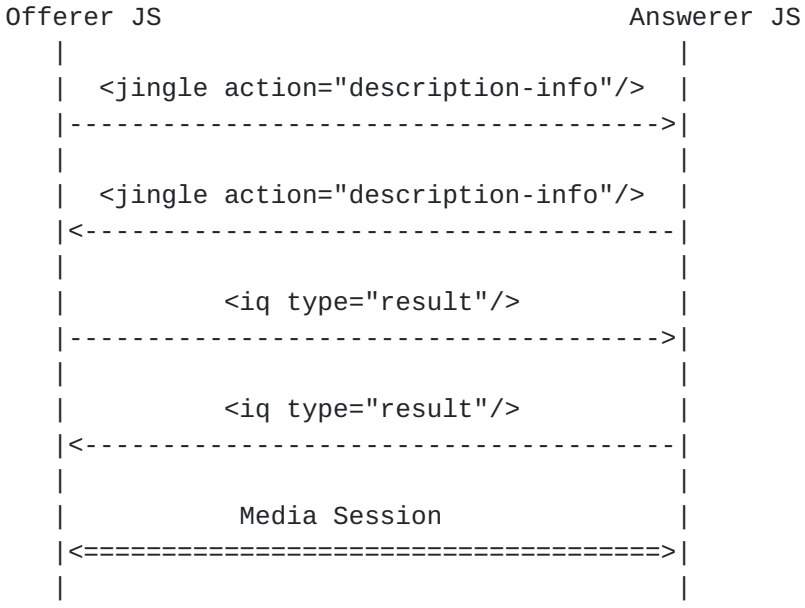

Figure 8: Exchange Description Information

Message details go here...

7. Security Considerations

TBD.

8. IANA Considerations

This document requires no actions from IANA.

9. Acknowledgements

The author would like to thank Kiran Kumar, Bert greevenbosch, Justin Uberti for the reviews and feedbacks.

10. Normative References

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-02 (work in progress), October 2012.

[I-D.jennings-rtcweb-signaling]
Jennings, C., Rosenberg, J., and R. Jesup, "RTCWeb Offer/Answer Protocol (ROAP)", draft-jennings-rtcweb-signaling-01 (work in progress), October 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[XEP-0166]
XMPP Standards Foundation, "Jingle", Dec 2009.

[XEP-0167]
XMPP Standards Foundation, "Jingle RTP Sessions", Dec 2009.

[webrtc-api]
W3C, "WebRTC 1.0: Real-time Communication Between

Browsers", Jul 2012.

Author's Address

Kepeng Li
Huawei Technologies
Huawei Base, Bantian, Longgang, Shenzhen
P. R. China

Phone: +86-755-28971807
Email: likepeng@huawei.com