

Aligning JSEP + BUNDLE

Justin Uberti, IETF 110

What's the issue?

- JSEP and BUNDLE specify contradictory ways of generating SDP offers and answers when bundling is presumed or accepted
- The most common implementation differs from both of these specs; while this is non-standard behavior, changes will have consequences to 3rd party apps

BundlePolicy

JSEP always offers to BUNDLE all m lines. However, it defines different 'policies' to control the ICE gathering behavior, which affects what happens when the remote endpoint doesn't support BUNDLE. **'balanced'** is the default behavior.

- **'balanced'**: candidates gathered only for first m section of each media type
- **'max-bundle'**: candidates gathered only the first m section
- **'max-compact'**: candidates gathered for all m sections

m lines without candidates cannot* be used with non-BUNDLE endpoints. These m lines are marked with a zero port in initial offers to ensure they are rejected.

Impact of BundlePolicy on offers (1 audio, 2 videos)

JSEP 'balanced' offer:

```
a=group:BUNDLE 0 1 2
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag0
a=ice-pwd:pwd0
m=video 10001 blah blah
a=mid:1
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:2
a=bundle-only
```

JSEP 'max-bundle' offer:

```
a=group:BUNDLE 0 1 2
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag0
a=ice-pwd:pwd0
m=video 0 blah blah
a=mid:1
a=bundle-only
m=video 0 blah blah
a=mid:2
a=bundle-only
```

JSEP 'max-compat' offer:

```
a=group:BUNDLE 0 1 2
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag0
a=ice-pwd:pwd0
m=video 10001 blah blah
a=mid:1
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10002 blah blah
a=mid:2
a=ice-ufrag:ufrag2
a=ice-pwd:pwd2
```

Differences between JSEP and BUNDLE

- Note the use of port zero to identify m lines that require bundling (which only happens with 'balanced' and 'max-bundle' policies).
- JSEP indicates this should only be done in initial offers, but BUNDLE says this should happen in answers and re-offers as well.
- Let's look at some examples...

JSEP vs BUNDLE offer/answer (1a, 1v, default)

Both specs generate identical initial offers, but BUNDLE sets the port of the bundled section to zero in the answer.

JSEP Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10001 blah blah
a=mid:1
a=ice-ufrag:ufrag2
a=ice-pwd:pwd2
```

JSEP Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10000 blah blah
a=mid:1
```

BUNDLE Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10001 blah blah
a=mid:1
a=ice-ufrag:ufrag2
a=ice-pwd:pwd2
```

BUNDLE Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

JSEP vs BUNDLE offer/answer (1a, 1v, max-bundle)

Same difference as in 'balanced'. Note the use of port zero in the offers due to max-bundle.

JSEP Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

JSEP Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10000 blah blah
a=mid:1
```

BUNDLE Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

BUNDLE Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

JSEP vs libwebrtc offer/answer (1a, 1v, max-bundle)

Libwebrtc (used by Chrome, Safari, others) mostly aligns with JSEP, but doesn't use port 0 in max-bundle offers. Port 9 is used instead, as no candidates are gathered for the second m= line.

JSEP Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

JSEP Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10000 blah blah
a=mid:1
```

Libwebrtc Offer: [[gist](#), [fiddle](#)]

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 9 blah blah
a=mid:1
```

Libwebrtc Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10000 blah blah
a=mid:1
```


Comparison

SDP Type	Behavior	Pros	Cons
JSEP	Port zero in initial offers, shared port in answers and subsequent offers	<ul style="list-style-type: none">• Safest behavior for offers to non-BUNDLE endpoints• Safest behavior for answers to existing WebRTC apps	<ul style="list-style-type: none">• Existing WebRTC apps may not understand a=bundle-only and ignore m= lines in offers
BUNDLE	Port zero in all offers and answers	<ul style="list-style-type: none">• Safest behavior for offers to non-BUNDLE endpoints• Simpler syntax	<ul style="list-style-type: none">• Existing WebRTC apps may not understand a=bundle-only and ignore m= lines in offers and answers
libwebrtc	Like JSEP, but port 9 in max-bundle initial offers	<ul style="list-style-type: none">• Safest behavior for offers to existing WebRTC apps• Safest behavior for answers to existing WebRTC apps	<ul style="list-style-type: none">• Unclear behavior when max-bundle offer received by non-BUNDLE endpoints

Breaking down the issue

1. what should BUNDLE/JSEP endpoints do in answers/re-offers (in any mode)?
2. what should JSEP endpoints put into initial offers for m= lines without candidates?
 - a. when in **max-bundle** mode (where only the first m= line has candidates)?
 - b. when in **balanced** mode with more than one audio or video stream (where only the first m= line of a given media type has candidates)?

Issue #1: Answers and re-offers

The BUNDLE answer will cause endpoints that understand BUNDLE but don't expect a=bundle-only/port zero in answers to incorrectly ignore the video m= line. Hard to know exactly how many applications will break, but almost all will be affected by the SDP change. The JSEP answer with m=video 10000 is admittedly less syntactically consistent, but safer.

JSEP Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10000 blah blah
a=mid:1
```

BUNDLE Answer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

Issue #2a: offers with a=bundle-only (1a, 1v, max-bundle)

The m=video 0 line in the JSEP offer (caused by use of max-bundle) may cause some WebRTC apps to incorrectly ignore it. On the other hand, non-BUNDLE endpoints that don't understand what libwebrtc is trying to offer may behave incorrectly (may generate ice-mismatch, may result in ICE failure, or may succeed).

JSEP Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:1
a=bundle-only
```

Libwebrtc Offer:

```
a=group:BUNDLE 0 1
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 9 blah blah
a=mid:1
```

Issue #2b: offers with a=bundle-only (1a, 2v, balanced)

Because there are two m= video m= lines, and we are using the 'balanced' policy, JSEP uses port zero for the second m= line. Libwebrtc on the other hand generates a max-compat style offer in this case. The overall problem is the same though, the remote side may ignore the lines marked with a zero port.

JSEP Offer:

```
a=group:BUNDLE 0 1 2
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag0
a=ice-pwd:pwd0
m=video 10001 blah blah
a=mid:1
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 0 blah blah
a=mid:2
a=bundle-only
```

Libwebrtc Offer:

```
a=group:BUNDLE 0 1 2
m=audio 10000 blah blah
a=mid:0
a=ice-ufrag:ufrag0
a=ice-pwd:pwd0
m=video 10001 blah blah
a=mid:1
a=ice-ufrag:ufrag1
a=ice-pwd:pwd1
m=video 10002 blah blah
a=mid:2
a=ice-ufrag:ufrag2
a=ice-pwd:pwd2
```

Data (source: [Chrome UMA](#))

BundlePolicy	Frequency
Balanced	85%
Max-bundle	7%
Max-compat	7%

App Type	Frequency	BUNDLE rate
Unified, 1 a/v stream	2%	94%
Unified, >1 a/v stream	2%	99.999%
Plan B	96%	99.9%

Observations

The data indicates:

- BUNDLE is extremely popular; essentially zero apps with >1 a/v stream and no BUNDLE
- Most apps (85%) using default ('balanced') BundlePolicy
- A meaningful fraction (X%) of apps use max-bundle or > 1 a/v stream with Unified Plan

From this we can conclude:

- Issue #1: changing the answer-with-BUNDLE behavior will affect over 90% of apps (== **high risk**)
- Issue #2a: changing the max-bundle libwebrtc offer behavior will affect some apps (== some risk*)
- Issue #2b: changing the >1 a/v stream libwebrtc offer behavior will affect some apps (== some risk*).
These apps do not talk to non-BUNDLE endpoints though (== low benefit).

* Firefox performed this migration 4 years ago successfully, albeit with some interop hiccups.

Recommendations

1. Switch BUNDLE spec to use JSEP behavior
 - Upside of BUNDLE SDP syntax change outweighed by risk to existing apps
 - Document that a=bundle-only in answers should be accepted but not generated
2. Investigate transitioning libwebrtc to JSEP behavior
 - Consider leaving **max-bundle** behavior as-is in libwebrtc; and change the name in JSEP to something else (max-bundle-safe?), to protect existing apps
 - Independently of this WG, attempt to migrate libwebrtc's **balanced** behavior to match JSEP. Re-evaluate path forward based on results.

Slightly off-topic, but perhaps worth discussing

1. BUNDLE doesn't provide explicit guidance for the situation where you get a=bundle-only for a m= line not in a BUNDLE group (although it does note it is unspecified).
2. This could happen when offering to a non-BUNDLE-aware endpoint that blindly copies media-level attributes in its answer.
3. Simple solution: add text to specify that BUNDLE implementations **MUST** ignore a=bundle-only attributes for m= lines not in a BUNDLE group.