What Developers Should Know About ORTC Versus WebRTC

ProgrammableWeb published an article recently about the availability of Microsoft Edge with ORTC API support which is part of the latest Windows Insider Preview release. Developers will now be able to build advanced communication applications on top of the Microsoft Edge browser via the ORTC API. Microsoft Edge is the only browser at the time of this writing that features ORTC API support. Chrome, Firefox, Opera, and Android support WebRTC 1.0 and the RTCPeerConnection API.

This article covers some of the differences between the ORTC API and the WebRTC 1.0 API, which is actually a set of interrelated APIs that includes the RTCPeerConnection API. This article also covers what ORTC API support in Microsoft Edge and other browsers may mean for application developers. Before covering the differences between the ORTC API and WebRTC 1.0 API, it is helpful to understand how ORTC came to be.

In early 2013, Robin Raymond, Hookflash CTO, and Erik Lagerway, Hookflash COO and co-founder, expressed some concerns regarding the use of session description protocol (SDP) in WebRTC/RTCWEB. Raymond published an in-depth blog post where he summarized his issues regarding SDP. Below are a few of Raymond's points taken from his blog post (Read the complete post).

- **Unneeded – "much too high level an API." -** The API should be lower level and wrappers can be created allowing simple access for

developers who would want higher level access.

- **Arcane Format – "legacy and problematic." -** "The SDP format itself is arcane and rooted in old world legacy reasoning."
- **Offer/Answer - "There's no SDP offer/answer needed."-** He says that managing streams with a solid API does not introduce a security threat and that "you are forcing all protocols in the future into an offer/answer model."
- **Incompatibilities** – Raymond wrote that "I dread the support nightmare with Open Peer when people bridge into SIP and we then receive no end of complaints about various incompatibilities between an Open Peer device talking to a SIP device; all because of junk crammed into SDP from these devices which isn't mutually supported."
- **Lack of API Contact** – Raymond says that "with an API, it's a contact. You don't change the contact arbitrarily because it has implications. An API can be extended, but with the current API you know exactly what you get thus you can predict behavior. SDP is not such. It can be changed arbitrarily and there's no guarantee the two will match."
- **Doesn't Truly Solve Goal of Compatibility to Legacy Systems** – Raymond says that SDP is actually a hindrance to compatibility and that SDP is not required for compatibility. Raymond also says that "I think the SIP vendors think if the browsers use SDP they will gain lots more compatibility. They won't. They are better off writing a JS library that talks the SDP they understand if they want SDP, rather than trying to mix/match browser SDP into the mix."

Shortly after publishing the blog post in March 2013, Raymond and Lagerway, with the help of a few others, submitted their first WebRTC Object API proposal to the Internet Engineering Task Force (IETF). In July 2013, the W3C ORCA Community Group was formed and in December 2013, the ORTC.org website was launched.

Today, Object RTC (ORTC) is a W3C community group supported by Google, Hookflash, Microsoft, and other leading technology companies. ORTC is not an official W3C specification at this time; it is an open project that aims to establish a standard for real-time communications (RTC). ORTC enables mobile endpoints to communicate with servers and web browsers using RTC capabilities provided by native JavaScript APIs and HTML5.

ORTC is often referred to as WebRTC 1.1 and is considered by many to be an evolution of WebRTC 1.0; it is not a replacement for WebRTC 1.0. In fact, several ORTC concepts have already been woven into the WebRTC 1.0 specification such as RtpSender and RtpReceiver. Google is also working on moving elements of the ORTC API into Chrome source code.

The key difference between the ORTC API and the WebRTC 1.0 API is that the ORTC API is a lower-level JavaScript API that provides the same components as the WebRTC 1.0 API while allowing greater flexibility than what is currently available in the WebRTC 1.0 SDP interface. Developers can use the ORTC API to implement advanced capabilities such as layered video coding, simulcast, scalable video coding (SVC), and more. These capabilities can also be implemented using SDP in WebRTC 1.0. However, WebRTC 1.0 communication capabilities can be more difficult for developers to implement. A JavaScript API like the ORTC API, provides greater access to more controls. In WebRTC 1.0, modifying the same controls would require browser source code changes. ORTC is also taking the approach of JavaScript shim libraries; for example, ORTC allows a JavaScript-based shim ("upshim") to be built on top of ORTC.js which would provide the same functionality of WebRTC 1.0 APIs, including the RTCPeerConnection API. It should also be noted that ORTC APIs do not replace WebRTC 1.0 APIs, but rather augment the existing SDP APIs in

WebRTC 1.0 which have been integrated into millions upon millions of devices.

Doug Mahugh, lead technical evangelist at Microsoft Open Technologies, Inc., published a blog post late last year in which he said:

> "ORTC leverages JavaScript to enable plugin-free real-time communications among web browsers, mobile devices and cloud technologies in a way that is familiar to website developers. The ORTC API is well suited to a "mobile first, cloud first" world because it supports advanced video features such as scalable video coding and simulcast. These advanced video technologies have proven difficult to support in an interoperable way within SDP in WebRTC 1.0. By contrast, utilizing these advanced video technologies within a JavaScript object API is more straightforward."

According to Justin Uberti, principal software engineer, WebRTC Project at Google, the next generation of WebRTC/ORTC is tentatively named "WebRTC NV" and will be a full convergence of ORTC and WebRTC 1.0. Applications will be able to program to WebRTC 1.0 (high-level) or object APIs. Developers will also have the option of bypassing SDP completely.

Choosing whether to implement ORTC and/or WebRTC 1.0 is not the only issue for developers of communication applications at this time. Developers of video communication applications must also carefully choose which video codec to support; the leading browsers currently support several different video codecs.

At the time of this writing, H.264 and VP8 are among the most commonly used video coding formats. Microsoft Edge currently supports H.264UC

(Microsoft's proprietary video codec) which is used by Skype services for enabling scalable video coding, forward error correction, and simulcast capabilities. Microsoft plans on adding H.264 support in the near future which will enable video interoperability between Skype and the FireFox browser (FireFox partially supports H.264) as well as the Chrome browser ("when H.264 support is added to its WebRTC implementation"). It appears that Microsoft Edge will not include support for VP8, a video codec currently supported by Chrome, FireFox, and Opera. However, Microsoft does plan on adding VP9 support to Microsoft Edge in the near future. Chrome, FireFox, and Opera already support VP9. Apple Safari currently supports H.264 and VP8 (must be installed separately).

Going forward, leveraging both the ORTC API and WebRTC 1.0 API may not be all that problematic for developers of communication applications. There are many communication service providers such as Sinch, Skype, and Twilio that will be providing SDKs that allow developers to easily build applications that support both. ORTC is backwards compatible with WebRTC 1.0 so it is not necessary to rewrite existing communication applications. It is also highly likely that the major browsers will eventually support ORTC, with the possible exception of Apple. It appears that the Apple Safari browser supports neither ORTC nor WebRTC and it is unclear as to what Apple plans on doing, if anything, when it comes to ORTC and/or WebRTC.

The more problematic issue for developers of communication applications appears to be with video codec support. With the major browsers choosing to support different video codecs, browser compatibility and interoperability for video communication applications may become a rather difficult proposition for developers. The IETF recently decided that both VP8 and H.264 are to be mandatory in WebRTC. However, most of the major browser vendors already support newer video formats like VP9 and a few

are working on adding H.265 support. The IETF mandate means that WebRTC web browsers must implement *both* VP8 and H.264 codecs. Developers that would like to build WebRTC compliant applications will also have to implement *both* VP8 and H.264 codecs.

Developers of communication applications should pay close attention to future web browser releases to determine when they can add new communication capabilities via ORTC APIs and support for new video formats to their applications. Communication application developers should also pay close attention to the work being done by the ORTC and WebRTC project groups. The work of these groups will most likely have a direct and profound impact on the future development of web browsers and communication applications.