WHITEPAPER

WHY A WEBRTC APPLICATION CONTROLLER

Quobis

EXECUTIVE SUMMARY

With WebRTC, a web service or mobile app can support real time communications services in a native way. Designing and deploying in production actual enterprise and carrier services goes beyond adding a few new pieces on its own, it can be time consuming and expensive, sometimes even writing off some of the WebRTC advantages.

One of the most experienced vendors in WebRTC arena, Quobis has developed a WebRTC Application Controller (or WAC) to manage and help integrating real time services into enterprise business processes and telco services.

WAC offers orchestration capabilities and a unified entry point to enterprise and telco networks, such as security or authentication flows, call dispatching, CRM, and so on; thus minimizing impact on those elements and extending their services to the new applications. WAC also isolates web app developers from browser variations (subtle, but critical, and continuously evolving) and hides behind a unified interface the different WebRTC platforms APIs, avoiding vendor lock-in and software complexity that would ensue otherwise.

Deployed in some of the biggest enterprise and telcos, WAC reduces costs, mitigates security concerns, and streamlines development and launch of production WebRTC based applications.

INTRODUCTION

This whitepaper covers the reasons why a new element called WebRTC Application Controller (or , WAC in short) is recommended to be included in any existing telco or enterprise architecture when you want to implement web communication services.

In the following pages we will introduce the problem of interconnection in WebRTC, including different legacy systems and the business opportunity behind this technology in service providers and tier-1 and tier-2 telcos. Later, we will explain the role of the WebRTC Application Controller as a border element in a IMS, NGN or core network of a telco and the top reasons to adopt this network element. Finally, we will mention how the WAC can be used in enterprise networks to boost new web services.

WEBRTC IN TELCOS AND ENTERPRISE

WebRTC is the next big thing in unified communications thanks to the new web browsers. These can manage voice and video communication in a native way, with no plugins, extensions, applications or updates to be installed.

WebRTC is a technology that involves the definition of different APIs and protocols, that are being standardized by W3C and IETF in a coordinated way. WebRTC technology was initially designed having browser-to-browser real-time communication in mind, but it is possible to be used in conjunction with different network elements to provide connection to IMS networks, enterprise platforms or the PSTN.

Google Chrome and Mozilla Firefox are the early adopters of this technology. Microsoft Edge has a specific implementation and

former Microsoft Internet Explorer and Safari have not yet announced native support for WebRTC, but there are different ways (via plugins, extensions or workarounds) to bring WebRTC capabilities to these platforms.

The independence from the platform and type of device, together with the fact that there is no need to install or update anything, is making easier the adoption by end users. WebRTC is independent from the device because PC, laptops, tablets, phones or smart TV (among others) can run a WebRTC enabled browser or even the WebRTC stack itself.

The use of WebRTC makes sense in enterprises and residential scenarios. There are lots of different use cases that can be useful. Together with the fact that the intelligence is in the browser, while call control and security remains in the network, and almost all devices are supporting them, the adoption is really fast and there are not collateral hardware costs.

Since Google Chrome and Firefox for Android support WebRTC, tablets have become a new endpoint for real time communications, taking into consideration that these devices have camera, are mobile and include a wide screen to extend collaboration environments that could not be deployed in smartphones. In addition, netbooks like Chromebook running browser-centric OS, are a new type of low-cost laptops that only run web browsers but can be a great opportunity to extend the reach of those services, using WebRTChrome with WebRTC is also running in smartphones with Android, so there is a possibility to extend WebRTC services to them. The main advantages are the independence of an application store

and, probably the most important, that customers do not need to upgrade anything.

It is possible to make WebRTC calls interoperable with other IP or legacy networks, by using WebRTC to SIP gateways. These elements have been released by Session Border Controllers, IP Centrex solutions and Media Gateway manufacturers during the last years. In addition, some popular open-source VoIP solutions are already supporting WebRTC and could be an interesting option for those scenarios where there are budget constraints.

This environment opens up big opportunities for new services for residential and corporate users. Residential users are demanding multimedia capabilities in different devices like new OTT services (similar to Line or Whatsapp) and vertical applications for e-health, online banking, etc. Corporate users are adopting WebRTC to solve communications needs in issues like BYOD, mobility and teleworking, with a seamless integration with their existing solutions.

These facts are convincing service providers and telcos to adapt their architectures to support WebRTC connections that involve the use of ad-hoc gateways and elements to manage the provisioning, authentication and billing of these new web services.

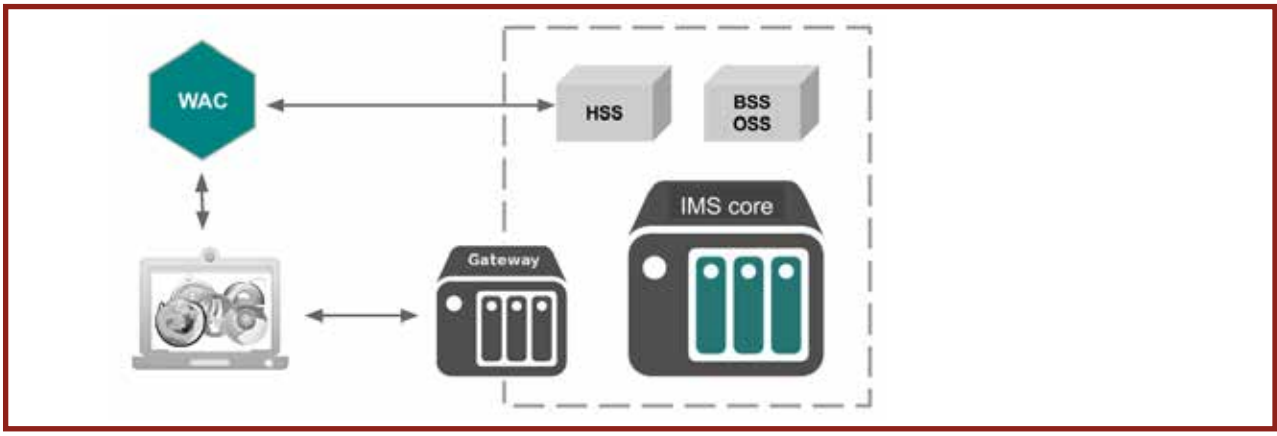
DESCRIBING THE WAC CONCEPT

In short, a WebRTC Application Controller (or WAC) is an indepedent network element that implements and manages all the new WebRTC-based applications that an enterprise or service provider wants to offer to their users. These web applications are served from the WAC, or existing web infrastructure, to the browser via HTTPS protocol like any other web application, enabling the browser to send media and signaling traffic to the customer network through the WebRTC gateway.

The WAC does not need to handle real-time traffic, but some WAC vendors can receive a copy of the signaling traffic to deal with information of the calls (monitoring KPIs, updating policy servers, etc). The following figure describes the role of the WAC in a real implementation, in this case a telco network.

A WAC can run on a dedicated server or virtual machine, which can be installed at the customer premises (as a new network element) or in the cloud (services providers may offer a WAC, or even a complete WebRTC application, as a service). It's important to mention that the WAC does not handle real time traffic but it enables the creation of new WebRTC-based applications.

The following sections describe in detail the role of the WAC in an enterprise or service provider WebRTC based service.



Hide complexity of user devices

WebRTC is designed to run in any type of device and platform with the unique need to use a browser that supports it. Unfortunately, today browsers like Chrome or Firefox for PC or Android devices are supporting different implementations of the W3C WebRTC API.

This makes WebRTC applications to behave differently depending on the type of browser and device. These differences are subtle, but can have big impact on the user experience, and require continuous commitment, as they evolve and change between versions and platforms.

In addition, Microsoft is not implementing the W3C WebRTC API in the Internet Explorer. They have announced they will support ORTC instead, adopted by Microsoft Edge. This forces web communication applications to develop a different version for legacy Internet Explorer.

There has been some announcements regarding the adoption of something called W3C WebRTC 2.0 API by all the existing browsers. The objective is to have some standard methods across all browsers in a near-future.

WAC helps alleviate these problems, hiding particular implementation details via the middleware called Abstraction Layer, with no need to make any change in the application code.

The WebRTC Application Controller is the element that is going to adapt each web application to the user platform depending on the type of device and browser that makes the request. This way, it is not needed to deal with different implementations and the complexity behind this, keeping complexity hidden. All the applications will work in an appropriate way independently of the type of device and browser that the user has.

As part of the QA activity, it's needed to perform tracks and tests of new browser versions to make sure any differences are addressed, and mitigate the impact to existing services. This is specially important for production services, where applications using the raw browser interfaces can be subject to disruption by a new feature or different implementation. By using the WAC, these problems can be eliminated.

Hide complexity of WebRTC gateway vendors

WebRTC does not specify anything for signaling, as was originally designed for peer-to-peer communications from browsers, and it was left up to each web application to decide on the best way to integrate signalling on their existing technology. As designing a signaling protocol requires quite specific experitse, there are different signaling standards promoted by entities and bodies like the IETF. Gateway vendors have chosen different ways to manage signaling, with protocols that go from standard based solutions, that are being adopted by some vendors and the open source community, to proprietary solutions based on SDK or proprietary APIs.

Standard based signaling protocols include SIP over websockets (SIPoWS) or XMPP over websockets, among others. SIPoWS is a friendly protocol for VoIP adopters who are already familiarized with it. It has been adopted by the most important VoIP Open Source solutions.

Proprietary and non-standard solutions or specific developments adopted by vendors include JSON over websockets

(or JSONoWS). This is a signaling protocol more friendly for web developers and a really flexible solutions. Another possibility is the usage of REST API (with events based on Websockets or long-polling) that is massively used in web applications. Finally, other approaches include Datachannels for signaling that is the mechanism that webRTC defines for exchanging data and reduce latency in critical environments.

These different signaling approaches, and others not mentioned, make WebRTC a complex architecture where an element like the WAC is needed to manage the complexity behind this different implementations.

The WAC helps to make WebRTC applications work with any vendor protocol, by offering a unified interface to the web app developer, rich enough to cover all the different requirements.

Manage the integration with existing services

It is really easy to implement a WebRTC basic demo service, as you can see from the different demos that are available on Internet. The big problem is when you want to create a WebRTC based service fully interconnected with your existing platforms. As we stated before, connection with the PBX (in case of an enterprise) or soft-switch or border element can be done via third-party WebRTC gateways, that are the network elements that will deal with signaling and media traffic, decoding and transcoding media and embedding traffic in SIP trunks. This traffic goes from the browser to the core network, so part of the problem is solved, but a solution is still needed to deal with authentication, authorization or accounting.

The authentication of users is one of the main tasks of the WAC. There are different possibilities open, like allowing anonymous calls where authentication is not needed (for instance, in some click-to-call solutions where only DoS protections are implemented) or the usage of the WAC as IAM (Identity and Access Manager) where it owns all the information of allowed users. Other possibilities are the use of telco authentication mechanisms (like AD, HSS or others) or, finally, rely on a third party via OpenID or OAuth (like Google, Facebook, Twitter, etc). In all of these scenarios the WAC is the element that must be configured to manage these procedures.

Authentication is just part of the problem. Authorization or the management of user privileges is another point to take into account. Service providers and telcos need to federate with their existing operation systems and policy managers. The WAC is the element that exposes some methods of an API in server mode for these systems and, hypothetically, can provide some connectors (in client mode) to ask OSS, BSS and other existing services about these features. As the WAC is the element that provides the WebRTC applications, this is going to be one that manages who has access to any application, service or feature.

Finally, accounting is another feature to be managed by the WAC. In this case, the WAC exposes some methods of an API to help BSS (billing systems) to ask about the usage of a WebRTC service. Part of this information can be accessed via the elements managing real time traffic (border elements, media gateways, etc), but the WAC provides additional information about the HTTP requests of the endpoints that can help to feed the BSS.

The WAC includes an orchestration capability so complex authorization and provisioning flows can be accommodated without impacting existing platforms or without requiring modifications on the WebRTC GW vendor software. With lots of available interfaces and services, and flexibility to develop additional ones, a WAC integrates legacy systems with WebRTC services, maintaining security and operational requirements.

Manage multi-tenancy services

Service providers and telcos are willing to offer new services to corporate customers so they can retain the revenues from this important market. In this scenario it's critical to have the possibility to offer services that are able to work in a multi-tenancy way. This means being able to run different instances of the solution to allow all the administrators of each company to manage different features of the service, with each enterprise totally isolated from each other.

The WAC is the element that is going to manage these different instances, allowing the service provider to offer applications for web communication with the branding of each company and providing the administrator users (of each company) with tools (via web user interface or similar) to manage users, privileges and services.

In addition, the WAC is the element that is going to provide detailed stats and call detailed records (CDRs) for the service providers and companies that are using these WebRTC applications. This element can provide connectors to feed well-known billing systems (BSS) and a service API to provide operation and billing systems with the methods and tools to manage multi-tenancy services (add or remove final users, add or remove groups or companies, request information about the usage of a service by a particular user or company, etc).

Mitigate web security concerns

Some new potential threats have started to appear with the use of WebRTC. This is on top of the already well known traditional VoIP attacks, many of which can also appear,, especially in those scenarios where interconnection is going to be present. It's important to take into account that a VoIP attack could cause an immediate financial damage for the attacked entity so preventing tools are almost mandatory.

This fact is new by comparison with other types of IT attacks, so it's important to take a look to the different types of threats and the protection methods to solve them. It's possible to group these traditional VoIP attacks in four types: denial of service (DoS), fraud, illegal interception and illegal control, where the systems or networks involved in the service can be attacked.

Ad-hoc attacks in WebRTC include access to physical devices. It is easy to figure out how risky would be to allow any web application to access user's webcam or screen (in case of screen sharing services) without asking for permission of the user.

Other attacks on WebRTC include cross-domain and DoS. This means that you can connect to a server whose domain is different from the domain you downloaded the code from. This gives the necessary flexibility to make this web communications useful, but it also could allow some kind of distributed Denial of Service attack, that should be prevented.

The use of ICE, TURN servers STUN candidates to avoid NAT problems can also create new problems. An attacker sniffing the signaling traffic will be able to know the private and public IP of each peer of the session. This information will allow, at least, the geographical position of the peers and can be an initial point for different types of attacks.

Security is one of the key features of the WAC, providing security mechanisms to prevent attacks and fraud in WebRTC sessions. It has been designed to deal with traditional attacks (illegal control, etc) and those related with the web.

Securing the network is a good option, but it is better to check if it is really secure. Some automatic tools can be used to simulate

common pattern attacks to check the behavior of the service in such conditions. Different pentesting tools can be used to implement a lot of common attacks and can also be easily customized. The service API of the WAC will help third party tools to evaluate the behavior of the services and prevent anomalies.

Lower the barrier to create applications

Different vendors are offering proprietary APIs or SDKs to help web developers to create their applications. While this is an acceptable option if a vendor is a long-term provider, there are different scenarios where a standard interface is preferred. Avoid vendor lock-in is one of the key points when you want to prevent spurious dependencies specially when you need to add or remove services or change vendors.

There are different standardization bodies and associations that are working to define standard methods in an API to integrate all the UC mechanisms involved in a RTC web application. GSMA, Open Mobile Association and others are some of the entities that have announced initiatives to promote an API. We are going to focus on ATIS orca.js initiative.

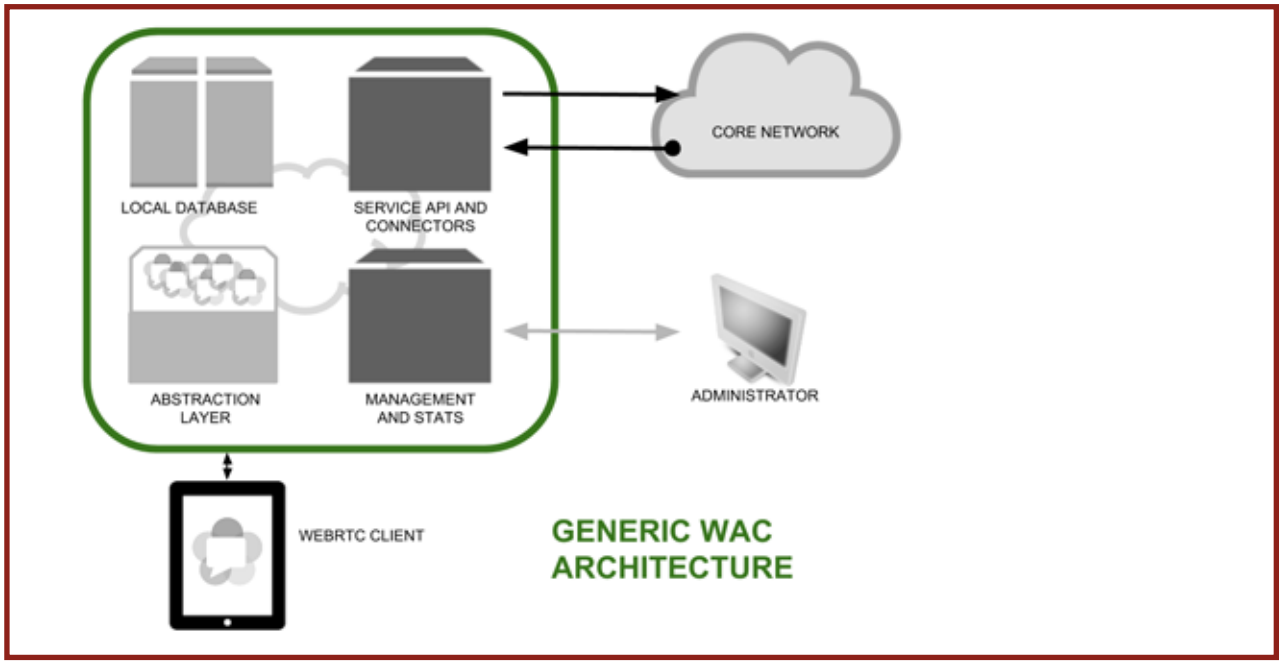
The Device Solution Initiative (or DSI group) in ATIS is defining orca.js as a set of methods to facilitate accessing the value of the network to devices and applications by reducing complexity in network APIs. The goal is to provide application developers with network-agnostic, client-side APIs that simplify the access to the network. This way, web developers can create applications via this API, include WebRTC functionalities in an easy way, with no need to have a deep knowledge of WebRTC standards and protocols.

ORCA.js initiative is supported by large vendors like Alcatel-Lucent and Ericsson and tier-1 telcos like AT&T, Verizon, CenturyLink or Sprint. We can foresee a way where the same network applications can work with any of these service providers and telcos with no adaptions.

The WAC, as the element that hosts all the WebRTC applications, is the right element to have a fully-compatible standard API to help third parties in order to develop their own applications. This WebRTC applications controller exposes this API and handles and hides all the complexity behind the network.

KEY FEATURES

The architecture or modules of the WebRTC Application Controller may vary from one vendor to another. In this chapter we will focus in those elements that are common to all the different implementations. We can group these features in six points.



Abstraction layer

All the WACs include an abstraction layer that helps to hide all the complexity of the different implementations of the WebRTC API in browsers. This element is critical due to the fact that is mandatory to make applications work, in a similar way, with all the type of devices and web-browsers. This feature helps web developers to avoid customizations related to the different types of endpoints.

Another point to take into account is the signaling complexity. Different vendors adopted different approaches, from standard signaling protocols to monolithic APIs or SDKs. Once more, it is important to prevent web developers to have problems with this and the WAC is the framework that make applications work with any gateway vendor with no changes.

Different WAC vendors may vary in the terminology but this abstraction layer may include also the modules that handle authentication, billing, user provisioning. etc. that can be implemented in the WAC or federated with existing systems via connectors or APIs that will be explained later.

Connectors with existing services

The WACs are designed to deal with different existing systems of the core network of a service provider or enterprise. The implementation of real WebRTC solutions in these scenarios makes necessary the use of systems to deal with issues like the authentication of users, service validation and/or exposure and billing reports. Common requests in terms of authentication demand validation with HSS (in case of IMS networks of telcos), active directories, LDAP or the use of OAuth or OpenID to leverage services offered by third-party Identity Providers such as Google and Yahoo. Some WACs include capacity to connect with different common vendors or solutions to manage authentication, so it is important to choose the one with the capacity to deal with the methods we need to implement.

These connectors could also be used to get the contact lists of the different users (for instance, from Microsoft Exchange platforms) or connect with policy servers

and service exposure manager to get information about the privileges of the users and information about the operation expected by the WebRTC services.

Service API

The connection with OSS and BSS is usually carried out via server mode API. This allows systems integrators who manage operation and billing systems (in the case of service providers) to have some methods to request information about the deployed WebRTC services or just to feed the WAC with updated information about the subscribers. For instance, OSS will use this API to add and remove users, groups, user privileges, etc. In the case of BSS, they use the API to request information about the usage of the WebRTC services via call detailed records the WAC is storing. Finally, WebRTC gateways can use also this API to ask the WAC about if a user that is demanding a WebRTC connection has been properly authenticated. If not, they will identify the traffic as an attack and will block it.

In most cases, due to the custom nature of most implementation and customizations of OSS, BSS, ACD or other internal systems, these elements are the responsible to federate with the WAC via the exposed Service API.

Developer API

The WAC, as the element that can act as web server, is hosting all the different WebRTC applications. This element is designed to allow third parties to use their own applications. For this purpose, it offers its own API in order to allow third parties (web developers) to adapt or create their applications, that can run in any device and architecture. This API has methods to manage basic communications setting and actions (make or receive a call, etc).

What makes the WAC interesting by comparison with other vendors strategy is the fact that this API is based in industry standards and avoids vendor lock-in of traditional gateway vendors. Different bodies and associations are defining telecom and enterprise APIs (OMA, GSMA, ATIS, etc) that are being adopted by gateway and application controller vendors.

These standard APIs guarantee that all the applications are going to work with no adaptations when you are using vendors that implement these methods. It's important to note that some vendors are making announcements about the support of a standard API but they make some improvement (additional methods) to provide extended functionalities. If you want to be sure that the applications is going to work with any vendor, you must limit to use the standard methods and calls.

Storage database

An element always present in the WebRTC Application Controllers is a database, that can be hosted in the same server or in an external element. A database is needed because WACs are called to manage services that require data storage. An example is the call detail records, where this information must be stored for a period of time. In fact, the web application can request data from the CDR to show information to the users (think in a WebRTC corporate endpoint where one of the possibilities is to know details of the last calls).

Chat history is another feature that he uses the database. It is great to have information of the previous messages via chat (including those

of previous sessions) before starting to write a new message. In addition, this database may be the responsible to store files that are transferred in the collaboration sessions.

Finally, the WAC can deal with authentication if this feature is not federated with existing systems, users and passwords can be stored locally in the WAC. Similar to this are the contact lists of the users that can be stored locally or connected with a Microsoft Exchange or others Network Address Book platforms. In the first case the WAC will use its own database to store all this information. In the second case, the WAC can keep a cached copy (kept up to date using appropriate NAB facilities) or delegate all requests to the NAB itself, depending on the specific use case.

Management tools and stats

The WAC includes a module to manage records and stats, where general and local administrators can get information about the behavior of the different web applications (concurrent calls ongoing, maximum amount of active users, etc) and, in the case of the system manager of the WAC, get detailed KPIs about the performance of the system (use of memory allocated, etc).

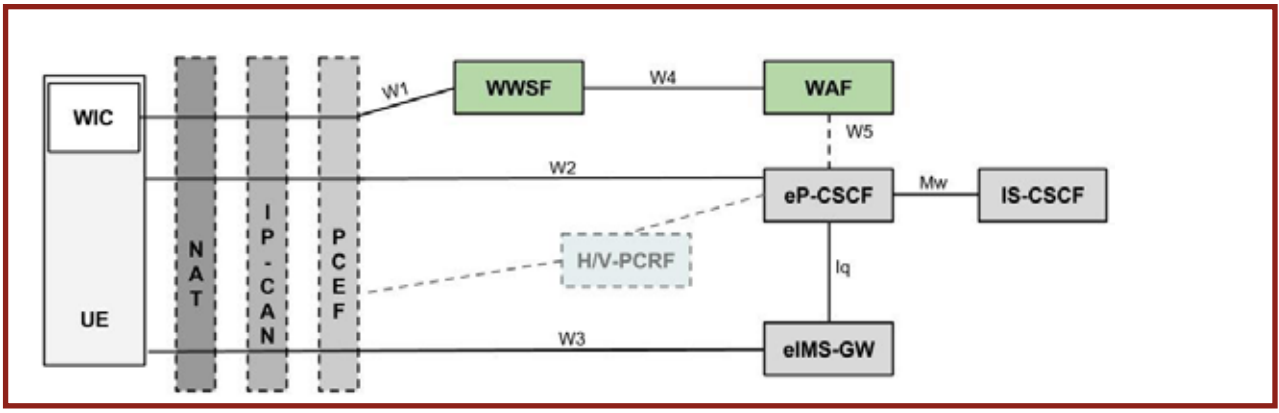
This is automatically done for all applications, without the developer having to worry or modify the application to collect quality or usage information.

Finally, it provides a user interface (usually based on web or CLI) to help all the different types of administrators to deal with the configuration of the services, from providing parameters to define a service to a specific user to remove a complete service or company, in the case of WAC managers.

In cases where the organization already has an accounting and auditing facility, the WAC data can be used as additional witness for validating service usage or billing, and help investigating any discrepancy.

THE WAC AS AN STANDARD 3GPP ELEMENT

The WAC is compliant with the 3GPP 'WebRTC access to IMS' architecture specified in TS 23.228 Annex U (Release 13). The following figure represents the general architecture proposed by the 3GPP, with the different elements:



The WAC plays the role of WWSF and WAF elements together:

WWSF (WebRTC Web Server Function).

It is the initial point of contact in the Web that controls access to the IMS communication services for the user, as it provides the WIC application for downloading to the browser. Among other features, the WWSF manages the allocation of authorized IMS identities to WIC and manages which authentication method applies. WWSF is just one of the various logical functions implemented by the Web Application Controller (WAC), and can be located either in the operator network or a third party network.

WAF (WebRTC Authorisation Function):

It supports the WWSF and eP-CSCF during the authentication and identity validation process. WAF is just one of the various logical functions implemented by the WAC. The 3GPP states that the WWSF can include WAF functionality when both elements are in the same domain.

The most relevant reference points or interfaces using the names of the figure 1 are:

- The W1 reference point is between the UE and the WWSF, and the HTTPS protocol is normally used. WAC typically support not only WebSockets (incl. its secure version) for efficient data exchange but also other fallback mechanisms (e.g. REST) to enable the communication in environments where middleboxes like HTTP proxies or firewalls might block some connections.

- The W4 reference point is the signaling interface between the WWSF and the WAF. The WWSF obtains an authorization token from the WAF which asserts the user's identity in the case of IMS registration scenario based on web authentication. This reference point is realized as an internal interface in the WAC.

- The W5 reference point is not specified and is implementation specific. In the case of different vendor solutions, the general approach with gateway vendors is based on a token-based authorization and identity mapping procedure is supported and pre-integrated, however the solution provides the building blocks to flexibly support other authentication, authorization and identity management mechanisms (such as OAuth, LDAP-based, SQL-based, REST-based, etc) integrating web application login and SIP authentication with the operator's IT infrastructure, OSS and HSS.

The WebRTC IMS architecture supports the following different IMS registration scenarios that might differ in the authentication method.

WIC registration based on web authentication

- From the WebRTC-enabled browser, the user accesses the URI to the WWSF to initiate an HTTPS connection to the WWSF and gets a token. The browser downloads and initializes the WIC from the WWSF.

- The WIC opens a connection to the eP-CSCF and tries to register, attaching the token from the WWSF. This validates the token against the WAF and maps the Web id with the SIP id.

- The eP-CSCF validates the token and forwards the register request to the S-CSCF, that responds with a 200 OK. The eP-CSCF sends the OK response back to the WIC

WIC registration based on IMS authentication

- From the WebRTC-enabled browser, the user accesses the URI to the WWSF to initiate an HTTPS connection to the WWSF and gets a token. The browser downloads and initializes the WIC from the WWSF.

- The WIC opens a WSS connection to the eP-CSCF and initiates a registration transaction with IMS via the eP-CSCF by sending a REGISTER request with the IMS credentials. This process leverages user credentials in HSS.

UNDERSTANDING THE ROLE OF THE WAC

This chapter includes the main reason why a WebRTC Application Controllers is needed when you want to deploy WebRTC applications in your network.

One box to manage WebRTC applications

There are different points to take into account in the implementation of WebRTC services, especially when we are working in a service provider or big enterprise network. These customers are going to demand services like multi-tenancy support, mechanisms to manage billing or other similar features needed for their business processes. In any case, both enterprises and service providers will need to deal with authentication, authorization, user management, service enablement and links to other existing services, so integration with OSS/BSS elements is going to be mandatory.

One possible strategy is to ask the WebRTC GW vendor to implement the required hooks and APIs on their product, and, at the same time, modify each one of the applications to comply with the evolving OSS/BSS requirements. Experience has proved that this approach leads to increased complexity and costs, and delay, and in many cases is unfeasible, as for example the WebRTC GW vendor can refuse to implement the kind of custom modifications that would be required.

A WAC is an element that is designed to manage all the complexity behind these implementations and it is a unique point to host all the web-based real-time communication applications and give access to the end users. In addition, it enables the compatibility with different devices, browsers (with their own WebRTC API implementation) and vendors.

Finally, it's a control point to keep the applications secure and to store all the information related with the usage of the services.

Add redundancy to critical web applications

The WAC is the easiest way to provide high availability for those scenarios where redundancy is important. This element does not deal with real time traffic, so it's not critical in terms of hardware requirements. This way, it's easy to be implemented on virtualized architectures where active-passive and active-active models are easier to achieve. The alternative could be the usage of the own WebRTC gateway HA capabilities, that not all vendors offer, as the manager of WebRTC applications. This option makes it more complex, in terms of costs, to deal with redundancy, especially in those scenarios where browser-to-browser calls are one of the use cases.

In addition, the WAC can also deal with routes to access different WebRTC gateways depending on their availability or with balancing protocols like round-robin or others. This means that the usage of WACs is the best alternative to achieve fully-redundant functional architectures.

Different license models available

Most of the vendor solutions that are designed to deal with unified communication sessions are licensed based on number of concurrent calls. This is related with real costs of hardware where memory, CPU and eventually DSPs depend on the number of sessions managed. The WAC is an element where scalability depends on the number of queries, that is a parameter that will vary with the number of endpoints (browsers) trying to access to real time web applications and the group of methods and connectors needed to fit the services in the proposed architecture.

As the WACs are elements that do not deal with real time traffic, they can be easily implemented on virtual machines, with no intensive needs in terms of capacity (processor and memory). A common way to license this element is based on the number of sessions or concurrent calls, in a similar way to WebRTC gateways. Telcos and companies are demanding this license model when they are implementing also the gateway.

Additionally, there are other possibilities depending on the number of applications, registered users or just active users in a period of time. Hosted and on premises models are valid and depend on the vendor strategy. Most of them are supporting both models trying to adapt it to the strategy of the customer.

Fits with any vendor or existing architecture

It's well known that different vendors use incompatible flavors of SIP protocol to manage VoIP communication. This boosted session border controllers as a mandatory element when interconnection is needed. WebRTC technology have similar problems in real implementation because all vendors is using a different (or slightly different) signaling approaches, so web applications are not working properly when you made any change in the architecture.

The WebRTC Application Controller, using an abstraction layer, provide a powerful mechanism to deal with all this different implementations, so it is the right tool to work in a multi-vendor architecture. In addition, via the service API, it provides a complete set of methods that can be used by the existing systems and elements to manage user privileges, authentication (HSS, Active Directory and others), operation, provisioning, billing, etc.

Security from web attacks, mitigating VoIP threats

As we mentioned before, there are different attacks that can compromise a WebRTC service, from traditional VoIP attacks to attacks focused on WebRTC as a web service (distributed denial-of-service, etc), that can block the access to the service or just cause a huge economic fraud in minutes

One of the primarily uses of the WAC is to provide security for those casualties that the SIP-based session border controllers are not providing, like preventing web services from DoS or DDoS in anonymous calls (a typical case of use of click to call buttons in public web pages), implement grey lists or blacklists to keep fraudsters out, deal with user privileges and strong authentication, etc.

Scalability and performance

The network traffic may grow by the usage of WebRTC services (video, etc) and some hardware upgrades may be needed. Third party WebRTC gateways are critical elements because they manage all real time traffic. Depending on number of sessions or transcoding capabilities, some hardware updates may be required. This is not the case of the WAC, as it only manages HTTP request from endpoints (browsers) and APIs to connect with existing systems, the requirements in terms of hardware are lower.

As the WAC can run on physical or virtual machines, it's also possible to build a high-availability architecture based on web balancers with an active-active behavior. But an active-passive architecture could be enough if you are not dealing with hundreds of thousand of concurrent calls, so the costs can keep reduced. For sure, when you implement a WAC in your network, you want the least amount of interruptions, so it is great to have all the possibilities in mind. The WACs are elements easy to deploy and configure, so the scalability and performance is going to be guaranteed.

One of the advantages of using a WAC is isolating the applications from changes on the backend. Adding a new WebRTC GW from the same vendor; swapping vendors; software upgrades; architecture redesigns or migrations; … all are totally transparent to the applications, as they would contact the WAC. Due to the simplicity of the WAC APIs, deploying new WACs or changing the web platform topologies would not affect applications either.

SOME EXAMPLES OF USE CASES

WebRTC can be used in different ways depending on the business model behind the service provider or enterprise who wants to deploy these services. Here we are going to use three examples on the use of the technology and different roles of the WAC.

Banking

A major global bank with more than 100,000 employees and market leaderin several G-20countries was looking to add Web communication capabilities in different scenarios, that are really critical for their business continuity.

The primary voice application for this company is the contact center. This undertaking is a massively important function for a bank, dealing with premium customers who want to manage their funds or e-banking users with problems to use the platform, where they only offered a traditional voice service until some months ago.

In addition to the call center, the bank needs support for their internal users, who are demanding new secure tools to communicate from different devices and platforms everywhere (full-mobility with BYOD capabilities). As a resume, the bank wanted to:

- Add multimedia capabilities to the users (video, chat, screen-sharing, co-browsing).
- Improve mobility and device support, keeping the standard level of security.
- Save money and employee productivity.

- Improve customer satisfaction.

After thorough evaluation of different options, this customer decided to include a WAC ni the architecture to help with the deployment of the different WebRTC applications (click-to-call, corporate endpoint, etc), managing authentication and security and helping with the integration with the contact center platform and all the corporate PBXs without adding security risks.

HealthCare

Health and insurance companies are looking for tools to reduce the expenses related with their operation. During the last years some companies are adopting communication tools to give an answer to their employees and doctors. These are demanding mobility tools while the customers are willing to reduce their travels to hospital.

A private health insurance company has several hundreds of thousands of customers, owns a bunch of hospitals and have thousands of doctors with different contract models (from autonomous or independent to full-time employees). They have though about the use of Web communications services as they were moving to the web most of the internal services during the last years (CRM, ERP,....).

The WAC is the element that is helping this health company to deploy and implement different services like:

- Give multimedia support to their contact centers (chat and video are fully operational with no need to change the existing legacy platform).
- Improve mobility and device support, allowing doctors and the rest of employees to use collaboration suites everywhere.
- Provide customers with a click-to-call solution, helping to connect with their doctors and hospitals, that could be the first-stone to implement a full-electronic health system that can bring the doctor to home.

This company needed a WAC to help with the deployment of the different WebRTC applications (click-to-call, corporate endpoints, etc). They needed a powerful tool to manage the different user privileges (from customers to employees), with security authentication and security tools. In addition, this is the element that helps with the integration with the contact center platform and corporate PBXs.

Service provider for the residential market

Service providers and telcos which are offering voice services for residential users are worried about the loss of market that over-the-top services have caused. During the last years they were looking for solutions to keep part of the revenues from voice services. That's the case of a traditional tier-1 telco who thought that WebRTC could help to launch some new services that were required by their users (to give mobility to traditional land-line voice services, to allow multiple devices to receive call, etc). Some of the services they have launched include:

- Web endpoint to allows users to receive and make calls from any device.
- Click to call buttons from their "yellow pages" sites, to help user (mainly SMEs) to receive calls.
- Videoconference capabilities in television, to make possible to receive calls in the TV and/or make video calls to friends when users are watching a TV program.

The WAC manages the different applications and profiles of the users providing tools to deal with authentication and accounting, and making all web applications work in any device and platform. This element allows this service provider to improve the portfolio and provide advantages like:

- Add multimedia capabilities to their users (video, chat, screen-sharing, co-browsing) that where using only voice solutions.
- Improve mobility and device support, answering one of the demands of the users.
- Improve customer satisfaction.

FAQS

1. Some gateway vendors provide interconnection with existing systems in a similar way the WAC. Is the WAC needed in these scenarios?

It depends on your needs. WACs are elements designed to deploy web applications in real environments, while WebRTC gateways are designed to deal with sessions. If your needs in terms of interconnection (authentication. billing, service provisioning, multi-tenancy support etc) are related with the deployment of application, the WAC can be mandatory. For further information, take a look to the chapter that explains the different roles of the WAC.

2. Should I use third party applications or choose the ones provided by the network equipment vendor?

Network Equipment Vendors (or NEVs) are offering WebRTC applications so this is an open possibility, but you must take into account that none of them are web developers. They create applications to show the potential of their devices and have no real interests in adapting or integrating these applications for the specific demands of the end customers. This way, it's better to adopt those vendors that offer a standard based API or SDK to develop web applications, so you have more options to choose the final developer for your applications.

3. How a WAC can help to give WebRTC support to devices based on iOS?

WebRTC is designed to run in any type of device and platform with the unique need to use a browser that supports it. Today iOS platforms are not ready to use WebRTC so there is no other solution to use a native app for this. The WAC has an abstraction layer that helps to hide all the complexity we have behind the different implementations of the W3C WebRTC API in browsers so, as soon as Apple launches WebRTC capabilities, WACs are going to add support for this implementation, making the apps work with no need to change code.

4. Support different gateway vendors is cool, but I will have just one, why do I need this feature?

WACs help to avoid vendor lock-in. In fact, they are cool if you are working or testing different vendor solutions because you can use the same web application with all of the them. If this is not your situation, think in the future, you may change the vendor, close agreements with third parties, merge/acquire a competitor, etc. It's always better to be independent to a specific vendor. In any case, these is just a feature of the WACs, if this doesn't make sense for you, take a look to the rest of benefits.

5. Is the WAC the element that manages multi conferencing?

No, multi-conferences are managed by MCUs, that are media communication engines that exist before than WebRTC. In fact, you can use native WebRTC MCUs, WebRTC gateways with MCU capabilities or SIP, H.323 or legacy MCU, where you need to convert all the traffic to SIP and traditional codecs. So you have different possibilities here..

CONCLUSIONS

This white paper summarizes the reasons why a new network element called WebRTC Application Controller (or , WAC in short) is recommended to be included in any existing telco or enterprise architecture when you want to implement web communication services. This is the best options to deal with all the complexity of real field deployment like managing all the different devices, and browsers and the interconnection with different gateways and internal systems of enterprises and service providers.

WAC offers orchestration capabilities and a unified entry point to enterprise and telco networks, such as security or authentication flows, call dispatching, CRM, and so on; thus minimizing impact on those elements and extending their services to the new applications. WAC also isolates web app developers from browser variations (subtle, but critical, and continuously evolving) and hides behind a unified interface the different WebRTC platforms APIs, avoiding vendor lock-in and software complexity that would ensue otherwise..

About Quobis

Quobis is a leading european company in the delivery of carrier-class unified communication solutions with a special focus on security, interoperability and identity management for service providers and enterprises.

Quobis is headquartered in Vigo (Spain) with partners throughout the world. Quobis is well-known as one of the leaders in the standarization and deployment of WebRTC technology after being involved in the industry firsts implementations and PoCs in more than 30 countries. Quobis is the creator of the first industry WebRTC Application Controller.

QUOBIS NETWORKS SL
Pol. A Granxa P260