# Quobis documentation center

*Release 5*

**17 Apr 2024 at 10:00**

# QUOBIS WAC

This page list of the subpages included in this site and serves as a navigation sitemap.

CHAPTER

ONE

# WHAT'S NEW

Version v5.11 released

**24th November 2023**

Quobis WAC v5.11 has been released today, these are the main features in this version:

- Web pushes to notify users of incoming calls, even when the browser tab is closed.
- Improved flow in the attended transfer mechanism

More information is available in the *Release Notes*.

Version v5.10 released

**10th October 2023**

Quobis WAC v5.10 has been released today, these are the main features in this version:

- New integration with Microsoft Azure: federated login and agenda
- Admins can now update the user's credentials via a Kitter endpoint
- Improved audio quality with automatic bitrate setup
- Performance improvements at the message broker and QSS services

More information is available in the *Release Notes*.

Version v5.9 released

**19th July 2023**

Quobis WAC v5.9 has been released today, these are the main features in this version:

- New graphs in the monitoring tool: concurrents calls and sent alerts
- New kitter endpoint and extended search capabilities
- Improved performance by removing the "vExpress" core architecture

More information is available in the *Release Notes*.

Version v5.8 released

**28th April 2023**

Quobis WAC v5.8 has been released today, these are the main features in this version:

- For on-premises installations, we now support Longhorn as a storage backend. That's the perfect fit for a Kubernetes-based install.
- Support to X.509 digital certificates, in order to authenticate users with national ID cards.
- Support to XEP-0363 XMPP extension to share files via HTTPS
- A lot of new monitoring dashboards, graphs and bug fixes.

As usual, more information is available in the *Release Notes*.

Version v5.7 released

**13th March 2023**

Quobis WAC v5.7 has been released today with automatic identity provider discovery and a new ringing strategy "simultaneuos reduced" for user groups. More information in the *Release Notes*.

Version v5.6 released

**13th February 2023**

Quobis WAC v5.6 has been released today with remote-party audio mute/unmute management and some updates on the media services. More information in the *Release Notes* .

Version v5.5 released

**12th January 2023**

Quobis WAC v5.5 has been released today as the first release in 2023. These are the most relevant features included in this version, along with a number of bug fixes and minor changes. The complete list with all the details can be found in the *Release Notes*:

- Meetings can now be recorded, in addition to one to one calls. An overlay text can also be added in the postprocessing for watermarking and security.
- In order to prevent DoS attacks, a new security mechamism has been implemented in the websocket connection
- A maximum and minimum default resources value can now be set for each pod to prevent system overloads and better scalability

Version v5.4 released

**22th November 2022**

Quobis WAC v5.4 has been released today. These are the most relevant features included in this version:

- New dashboards to monitor the status of internal services
- Performance optimization and a new metric server to collect resource metrics from the Kubernetes infra.

along with a number of bug fixes and minor changes. The complete list with all the details can be found in the *Release Notes* .

Version v5.3 released

**26th September 2022**

Quobis WAC v5.3 has been released. These are the most relevant features included in this version:

- Users are now notified of missed calls that have been received while they were offline.
- A new REST API to setup the ringing strategy of a user group.

The complete list with all the details can be found in the *Release Notes* .

Version v5.2 released

**15th July 2022**

Version 5.2.0 of Quobis WAC has been released. The complete list with all the details can be found in the *Release Notes* .

Version v5.1 released

**6th July 2022**

Version 5.1.0 of Quobis WAC has been released. The complete list with all the details can be found in the *Release Notes* .

Version v5.0 released

**Vigo, 25th March 2022**

A new **major** version of the Quobis Communication Platform, named v5, has been released today and considered as a General Availability version. It includes a number of new features, changes and bug fixes. Version v4.3 is no longer supported as a result of this publication. All customers are encouraged to update to v5 as soon as possible.

These are just a few highlights from all the improvements in this version, you can find the all of the updates in the release notes linked below:

New authentication system

A new service has been added to improve login process and user account management. Previously, users needed to be provisioned by the administrator or exist in a third party authentication system. Now, users can sign-up by using a simple web interface, and also change and recover their password via a web interface.

On the other side, login templates can be customized for each of the login sections (new user, lost password, change password), additional security policies can be implemented (ex: forbid a user login after several invalid login attempts) and the integration of new authentication oAuth providers is now much easier.

There are much more new features and bugfixes included in the version. You can check the complete details in the release notes for all the products included in this version: Quobis wac v5.0 release notes

CHAPTER

TWO

# OVERVIEW OF QUOBIS WAC

The term **WebRTC Application Controller (wac)** has been coined by Quobis after our experience deploying WebRTC projects in service providers and enterprises all around the world.

**Quobis wac** is a solution developed by Quobis that allows telephone operators, CSPs (Cloud Service Providers), MVNOs (Mobile Virtual Network Operator) and large enterprises to create real time, multimedia and unified communication applications. These applications are used on web browsers, mobile browsers, mobile devices and embedded devices as well.

Why do I need a Quobis wac

Web and mobile applications are usually built on top of application servers or web frameworks, such as JBoss, NodeJS, PHP, etc... When it comes to real time applications delivering live audio and video, there are a large number of *functionalities* that need to be implemented and that are not present at generic application servers.

> Quobis wac fills these gaps that you need to take a application prototype into real production in a secure and scalable way

Powerful SDKs

Thanks to its *powerful SDKs* and flexible configuration options, Quobis wac supports a large number of business cases, from a simple click-to-dial button to advanced scenarios like UC collaboration, unified messaging, remote contact-center agents, video customer onboarding, etc.. in a number of verticals such as banking, health, logistics, retail, etc...

Standards based

Quobis wac is standards compliant and has been designed and developed by engineers who participate in WebRTC standardization forums like W3C, IETF, 3GPP, SIPForum and GSMA. It also has been designed as a carrier-class solution with special focus on scalability, reliability and interoperability with existing UC platforms and networks.

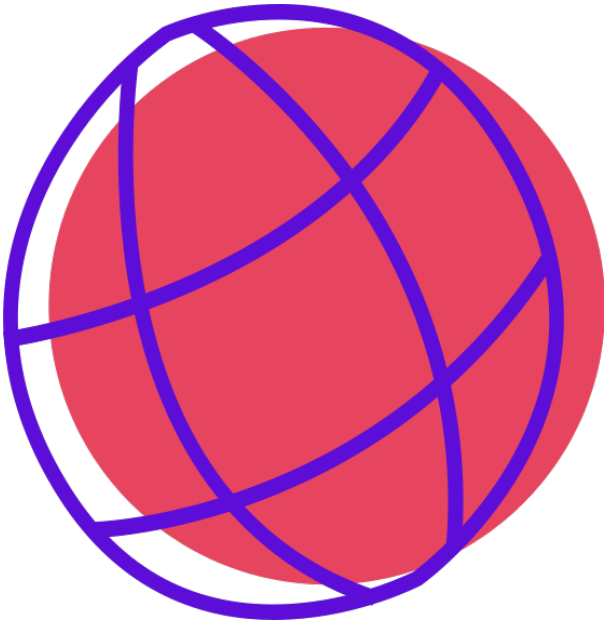
Cloud native

The Quobis Communication Platform can be implemented on-prem, Private and Public Cloud environments. It is deployed as a cloud native, containerized application, orchestrated and managed by Kubernetes.

## 2.1 About WebRTC technology

Quobis product family has its roots in the to WebRTC technology, which is a free, open project that enables web browsers with Real-Time Communications (RTC) capabilities via simple JavaScript APIs. The WebRTC initiative is a project supported by Google, Mozilla and Opera and its mission is to enable rich, high quality, real time applications to be developed in the browser via simple JavaScript APIs and HTML5, across multiple platforms. Nowadays, WebRTC technology can also be used on smartphones and also other embedded devices.

## 2.2 Features



Quobis wac provides a number of functionalities that boost the development of real time applications. The most relevant functionalities provided by Quobis wac are detailed in the list below:

HQ Voice and Video

Quobis wac is able to handle the setup of *real-time voice and video sessions* with different configurations, ranging from traditional telephony-like one to one calls to multiparty calls, meetings, personal rooms and other complex scenarios. Quobis wac also supports screen sharingand quality control methods. Its SDK come in different flavours for Javascript, Android, iOS and C++.

User management

Quobis wac handles the management of both registered and guest *users*, its capabilities and its classification into domains, supporting multiple tenants into the same deployment. Users can be classified into *user groups* to create teams, departments, etc... which can also share commons contacts as shared *phone books*.

Identity management

Quobis wac provides and internal authentication system to authenticate the subscribers, but it also can delegate the authentication of the users into a third party element that supports OAuth2 protocol, such as Microsoft Active Directory. Subscribers can have several identities and user a number of different authentication models.

Permissions and capabilities

Users are entitle to do different actions depending on the permissions, capabilities setup by the system administrator via a flexible REST API

Contact management

Quobis has an internal Network Addressbook (NAB) that can be used to manage the agenda of the subscribers. It takes into account that subscribers and belong to different domains, organizations and groups, so a number of combination of private contacts, shared contacts and group contacts is already available. This NAB can also be synchronized with external systems via a REST API.

Recording

The *recording* capability allows the implementation of use cases where there is a requirement for recording every interaction in a secure and encrypted way.

SIP interconnection

Beyond its internal signaling server, Quobis is able to connect to existing SIP and IMS networks with flexible SIP schemas, from a SIP trunk to complex registration scenarios and SIP manipulation rules. That enables a range of innovative applications to extended traditional telephony applications into the cloud.

Conferencing

Beyond one-to-one calls, Quobis wac allows the setup of multiparty audio and video rooms with fine-grained functionalities, such as public conferences, private conferences, one-use conferences, etc... depending on the use case, billing strategy, etc...

Messaging

Beyond audio and video communications, Quobis wac provides a messaging server to support use cases such as a basic chat, multi party chats, persistent chats, etc... That messaging capabilities can be combined with the user management capabilities as well

Notifications

Quobis wac handles the logic needed to properly notify users of incoming calls, new messages, new groups and other events, both for subscribers using a mobile device (via a push notification server) but also browser-based notifications.

Interoperability

Quobis wac has been designed with a open interoperability mindset. That means that part of the functionalities provided by Quobis wac can be replaced by third party elements without impacting the rest of functionalities. This is specially important when customers wants to integrate any existing WebRTC element such media servers, WebRTC-to-SIP gateways, etc...

Integration

Most features are available through REST APIs so easy OSS/BSS integration and service provisioning

On-prem or cloud

Quobis wac is a software-only solution that can be *deployed* in standard bare-metal server or in a Docker and/or Kubernetes environments, ready for cloud providers but also traditional on-premises deployments.

And much more!

There are also a number of low-level functionalities that are described in detail in the following subchapters that make Quobis Communication Platform a quite unique solution to deliver complex applications to life in a very short time frame in a trusted, secure and scalable way.

## 2.3 Application development with Quobis SDKs

Services providers and enterprises can develop their own WebRTC applications and enhance existing ones using Quobis wac SDK's, leveraging of all the Quobis wac *functionalities* like authentication, user management, contacts, etc...

These SDKs can be also customized for service providers

that want to expose their own set of SDKs to sell their services to small enterprises and developers to compete with existing SaaS providers of WebRTC APIs.

Every application developed with Quobis SDK requires connection with a Quobis wac, as some of the features are not implemented on the browser but on the Quobis wac. The applications and the Quobis SDK libraries can be hosted at the Quobis wac or at the customer own servers.

There are four SDKS available as of today, documentation and details are explained online at http://doc.quobis.com, with a set of tutorials and guides to help you getting started and develop your first application.

- Javascript SDK to develop web HTML5 applications
- Android SDK to develop mobile apps for the Android operating system
- iOS SDK to develop mobile apps for the iOS operating systems

img/multiplatform-composition.png

## 2.4 Interfaces and APIs

This section explains the main protocols and interfaces supported by Quobis wac and the connections points to other network elements. There are three main interfaces that connect a Quobis wac instance with the rest of the network and/or Internet:

***Quobis SDKs***
: These are a set of software development kits (in short, SDKs), available on different programming languages, that allow developers to create their applications and leverage all of the Quobis wac functionalities using WebRTC standards. Quobis clients are the applications that use the Quobis SDKs -in the browser on in mobile apps- to provide modern services such as audio and video calling capabilities, in-app messaging, multiparty video conferencing, etc...

***SIP interface***
: Beyond app-to-app or browser-to-browser communications, Quobis wac supports SIP protocol to interconnect with existing telephony and VoIP networks, both in enterprise and IMS-based service provider networks. That opens up a new range of use cases to make browser-to-SIP calls and app-to-SIP calls in addition to standard WebRTC-to-WebRTC calls, bridging both the telephony world and the internet world. That means that customers can extend the existing services provided on regular phones, PBXs, call-centers and other SIP-platforms and take them to websites and mobile applications.

***REST API interface***
: A powerful and flexible REST-based API is provided for OSS/BSS integration, system administration, user provisioning system configuration and monitoring. This REST API can be easily and securely consumed from third-party systems but also as from web and mobile applications, complementing the QuobisSDKs. The most common uses for this API are user provisioning and management, address book integration, call detail records collection, authentication and usage.

## 2.5 Network architecture reference

**Quobis wac** can be deployed in a number of different ways depending on the use case to fulfill. Generally speaking, Quobis wac usually sits in the edge of the network, in close collaboration with other network elements both on the telephony area (such as PBXs, soft switches, SBCs, etc...) but also on the IT area (web-servers, web-proxys, load balancers, firewalls, etc...).

The following picture describes a basic deployment of a Quobis wac. Please note that there is no "right" architecture as it's highly dependent on the use case. Quobis's presales team provides professional services to help in the design of the most appropiate architecture in terms of security, scalability, etc...

The main elements of this network diagram are:

**Quobis wac**

Usually sitting in the edge of the network, this element is the core of the real-time applications to be deployed.

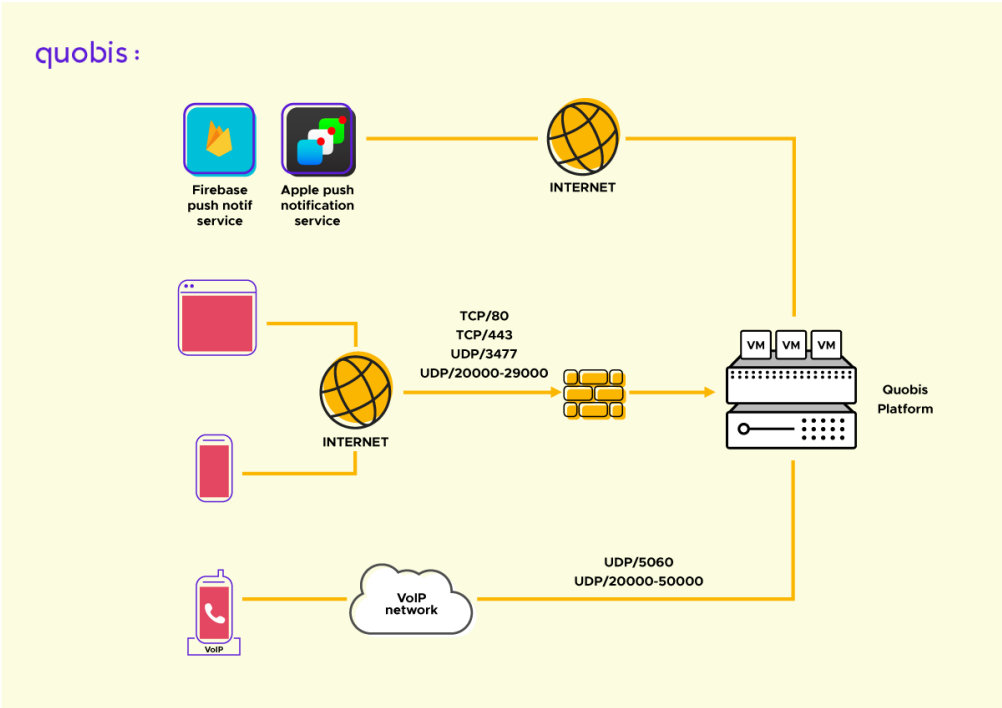
**Endpoints**

The WebRTC applications that are downloaded into the web browser (in the case of HTML5 applications) or deployed into a physical device (mobile phone, tablet, etc...) as a native application. From the point of view of the subscriber or end-user, this is the only application that he/she will need to use. Quobis applications have HTTPS and Websocket connectivity with the Quobis wac.

**SIP network**

This block is present when we need to connect to a existing PBX, ACD, softswitch, SBC or any other *SIP element*. Quobis wac has built-in SIP functionalities and thus can route call to the SIP network, via a standard SIP trunk or via a SIP REGISTER method. SIP authentication and *SIP credentials management* are also supported.

**IT network**

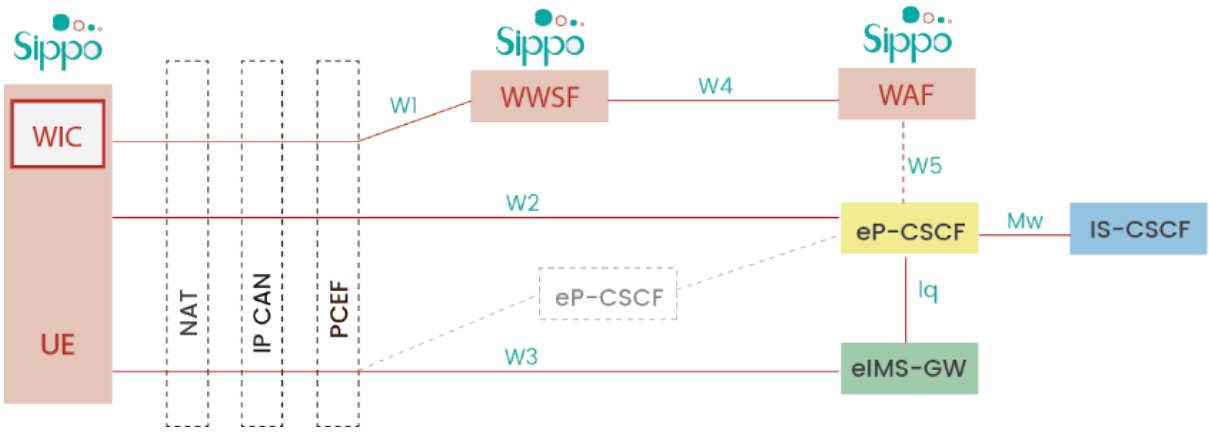



This block is present when Quobis wac needs to interact with existing elements on the customer network for user authentication, user provisioning, CDR collection, addressbook integration, access control, monitoring, etc. . .

## 2.6 Compliance with standard 3GPP architecture

The Quobis AS with the 3GPP 'WebRTC access to IMS' architecture specified in TS 23.228 Annex U (Release 13) and TR 23.701.

The following figure represents the general architecture proposed by the 3GPP, with the different elements provided by Quobis: WIC, WWSF and WAF




**WIC (WebRTC IMS Client)**
Is an application using the WebRTC extensions specified in WebRTC 1.0 spec, as any of the WebRTC apps built using QuobisSDK

**WWSF (WebRTC Web Server Function)**
Is the initial point of contact in the Web that controls access to the IMS communication services for the user, that is one of the features of the Quobis wac

**WAF (WebRTC Authorization Function)**
It supports the WWSF and eP-CSCF during the authentication and identity validation process.

**eP-CSCF (P-CSCF enhanced for WebRTC)**
Provides the signaling interworking for WebRTC clients access to IMS, while acting as the P-CSCF towards the IMS core via the Mw reference point.

**eIMS-AGW (IMS Access GateWay for WebRTC)**
Provides the media plane interworking capabilities such as DTLS-SRTP to RTP conversion, transcoding, and NAT traversal support including ICE.

## 2.7 GDPR information

Quobis solutions doesn't store or hold any personal customer data. Any other collected data is not correlated with any particular customer. It's up to the system administrator therefore to determine what data to gather from the customers when defining the context variables to use in each engagement.

CHAPTER

THREE

# QUOBIS WAC RELEASE NOTES

## 3.1 v5.12.0

Released 22/03/2024

### 3.1.1 New features

- Support to scenarios where we need to register the WebRTC users into a external SIP network (ROAD-5148)
- Allow recording of calls/meetings with more than 9 participants (ROAD-1380)
- New endpoint the returns the count of unread XMPP one-to-one messages per user (ROAD-5040)
- New permission "receiveCalls" that allows to block/unblock incoming calls to a given user (ROAD-5191)
- Added pagination to GET endpoint in /api/users at Kitter service (ROAD-5185)
- Support to connect to an Azure instance when Quobis WAC sits behind a HTTP proxy (ROAD-5028)
- Added a retry mechanism in case that a video recording processing fails (ROAD-5351)
- Added a job-scheduler to setup task in the future (ROAD-5339)
- Optimization of the avatar uploading process (ROAD-5210)
- Set a default value for new domains to delete group messages (ROAD-4954)
- Verify the working status of the events "users.registered" and "users.unregistered" (ROAD-5174)
- Check that an email is valid before storing it in the database (ROAD-2399)
- Added a liveness/readiness endpoint to Kitter service (ROAD-5422)
- Added a liveness/readiness endpoint to RabbitMQ service (ROAD-5422)
- Support to service deployment with Helm charts (ROAD-2021)

### 3.1.2 Bug fixes

- Conference chat are not being deleted after the conference is done (ROAD-3104)
- Upgrades of XMPP server do not migrate hosts' configuration files (ROAD-4332)
- There are different modules activated when provisioning a new domain than quobis.cfg.lua (ROAD-4971)
- The user is created but a "Failed to send execute actions email : 500 Internal Server Error" messages is returned (ROAD-5066)

- wac-core disconnects from RabbitMQ with an error log "406 PRECONDITION-FAILED" (ROAD-5068)
- Cannot log out a user with single-session configuration when the mobile app is running in the background (ROAD-5137)
- qss-invites-rooms memory consumption increases in some circumstances (ROAD-5177)
- Parameter "sort_by" of old users endpoint is not working (ROAD-5196)
- Endpoint for getting permissions does not work correctly for the admin when reading another user's permissions (ROAD-5201)
- Cannot access Azure personal contacts after deleting an user and recreating it again (ROAD-5214)
- Can't select Azure ADFS contacts when there are personal or domain contacts (ROAD-5236)
- The processing of recorded videos using H264 codec is not correct (ROAD-5381)
- AudioStatus unhandled error when trying to notify external peers (ROAD-5456)
- Azure contacts: Cannot read properties of undefined 'reading length' (ROAD-5526)
- Added missed parameters "azureOrgUsers" and "azurePersonalContacts" (ROAD-5241)

### 3.1.3 Changes and deprecations

- Reduce the font size of the video recording watermark text (ROAD-3374)
- Replace hardcoded paths so we can now deploy the services as subdomains instead of paths (ROAD-5323)
- Upgrade wac-core NodeJS version from 16 to 18 (ROAD-5150)
- Open only one TCP connection to message broker and per pod (ROAD-5288)
- Remove resolver usage from meetings service in wac-core (ROAD-2264)
- New IPC library for AMQP connections with RabbitMQ. Now all messages are acknowledged (ROAD-3791)
- New MongoDB migration system in wac-core (ROAD-5125)
- Optimized index for admin conference log reading (ROAD-5142)
- Updated RabbitMQ version to 3.12.13
- Changed default value avatarBase64Persistence to notReadAndNotSave to avoid using base64 (ROAD-5216)
-

## 3.2 v5.11.4

Released 22/01/2024

### 3.2.1 Bug fixes

- Include the phonebook name of the caller when sending a push notification to a mobile subscriber (ROAD-4464)

## 3.3 v5.11.3

Released 20/12/2023

### 3.3.1 Bug fixes

- Set WAC_IPC_NO_ACK to false on services connecting to message-broker

## 3.4 v5.11.2

Released 05/12/2023

### 3.4.1 Bug fixes

- Attended transfer fails when two sequential transferred calls are established with the same target (ROAD-5156)

## 3.5 v5.11.1

Released 28/11/2023

### 3.5.1 Bug fixes

- Allow to make an attended transfer when there is more than one media server (ROAD-5153)

## 3.6 v5.11.0

Released 24/11/2023

### 3.6.1 New features

- New attended transfer flow: the target only receives one single call from the transferror (ROAD-4771)
- Added parameter xmppArchiveDeletion for configuration of archive cleanup time in XMPP server (ROAD-4942)
- Added parameter mucLogExpiresAfter for configuration of MUCs messages cleanup in XMPP server, defaults to 6 months (ROAD-4952)
- Added timestamp Y-m-dTH:M:SZ to XMPP server logs (ROAD-5089)
- Support for XMPP mod_muc_offline_delivery to allow push notifications from MUCS to users who are not currently connected (ROAD-4462)

### 3.6.2 Bug fixes

- In-conference chat messages and MUC are now deleted after the call is finished (ROAD-3104)
- The wrapper does not disconnect from the dispatcher when it has a connection problem (ROAD-4346)
- Fixed error with the conference-log table in some specific cases to SIP destinations (ROAD-4786)
- Content-type header is missing in avatar images and chat images (ROAD-4928)
- Many "recipient unavailable" appear in the prosodyarchive database (ROAD-4939)
- The parameter "getCurrentInvitationsByUsername" is not indicated in the QSS templates (ROAD-4970)
- The ID is not generated during sip mapping and generates a 500 internal server error (ROAD-5072)
- Longhorn doesn't free up pv space when data is deleted from it (ROAD-5090)
- Activate module disk by upload files XMPP (ROAD-5095)
- Alerts API canceledSince filter doesn't work (ROAD-5107)

### 3.6.3 Changes and deprecations

- Removed deprecated services that are not needed in wac-core
- Update loki and promtail versions to 2.9.2 (ROAD-5088)
- Set the correct XMPP Prosody modules: removed BOSH, disco; added REST (ROAD-4826)
- Remove vexpress config from erebus (ROAD-4974)

## 3.7 v5.10.1

Released 29/11/2023

### 3.7.1 Bug fixes

- The ID is not generated after the setup of a SIP mapping, giving an 500 internal server error (ROAD-5072)

## 3.8 v5.10.0

Released 10/10/2023

### 3.8.1 New features

- Added integration with Azure organization users ("azure_organization_users") (ROAD-4613)
- Added integration with Azure personal contacts ("azure_personal_contacts") (ROAD-4615)
- Added new parameter "oneLineObjects" to log JSON objects without line breaks, activated by default. Can be set to "0" if multi-line logs are needed (ROAD-4905, ROAD-4690)
- Added "credentials" parameter in Kitter "patch" payload to change user credentials (ROAD-4654)
- Added the option of deploying the MongoDB database in its own unique namespace (ROAD-4446)
- Added logs on user created and deleted in the "users" endpoint (c232860)

- Include the phonebook name of the caller when sending a push notification to a mobile subscriber (ROAD-4464)
- Clean-up dispatcher database entry after a call is made (ROAD-3343)
- Logging slow queries in PostgreSQL database (ROAD-4281)
- Block desktop apps access for unsupported versions (ROAD-4770)
- Serve file for deeplink support in iOS application (ROAD-4763)
- Configure cache headers for user uploaded files (fs_location) (ROAD-4618)
- Improve performance of user conference log and login queries (ROAD-4588)
- Configure Nginx cache headers for user uploaded files (/fs) to improve load times (925a82f)

### 3.8.2 Bug fixes

- "GetCurrentInvitations" message without content that affected clients using Callkit (66a70d6e)
- Conference number shown in Grafana is wrong (ROAD-2800)
- Logged user number shown in Grafana is wrong (ROAD-4678)
- There is no log when a user is created or deleted if all went OK (ROAD-3136)
- registerXmppPushToken service call fails with "database operation failed insert" (ROAD-4219)
- The "acceptedInvites" collection remains stored in the QSS database (ROAD-4247)
- Unable to deploy sippo-storage due to a serviceAccount wrong value (ROAD-4293)
- Wrong certificate assignment into xmpp-server (ROAD-4295)
- Remove "" in all aliases from Kamailio config ROAD-4301
- Asterisk RURI domain is hard-coded in the Kamailio config template (ROAD-4315)
- The container for qss-io, qss-invites,qss-audiostatus and qss-calls is named QSS instead of its actual name (ROAD-4333)
- Avoid collection scans in the dispatchger service to prevent slow querys (ROAD-4366)
- Unable to upload files from mobile apps (ROAD-4387)
- Grafana deployment fails at the end of the process (ROAD-4445)
- The pvc hosts and var-lib are created when deploying the XMPP server (ROAD-4619)
- Wrong behaviour when calling to a sequential ringing-group and the last candidates has more than one acve session (ROAD-4635)
- Incorrect memory management after deleting an user that leads to a memory peak (ROAD-4636)
- Incorrect managent of duplicated phone numbers when creating new users (ROAD-4667)
- Resources from MongoDB pods are not applied to database-cluster-operator deployment (ROAD-4669)
- Wrong response code "603" instead of "200" in accepted calls to SIP destinations (ROAD-4775)
- Last user of a reduced simultaneous ringing group does not received the incoming calls (ROAD-4823)
- Mailer is not sending proper EHLO command in SMTP dialogue (ROAD-4835)
- Attended transfer to PSTN users with parallel SIP is not working (ROAD-4840)
- The sfu-wrapper doesn't reconnect to the dispatcher when it loses connection (ROAD-4841)
- wac-core service consumes too much CPU when migrating sip mappings to PostgreSQL (ROAD-4918)

### 3.8.3 Changes and deprecations

- Upgrade to Longhorn v1.4.2 (ROAD-4447)
- Upgrade to MongoDB 6.0.4 (ROAD-4291)
- Upgrade Grafana version to 9.5.5 (ROAD-4538)
- Upgrade to kube-state-metrics 2.7.0 (ROAD-4292)
- Upgrade Kamailio from 5.5.5 to 5.6.4 (ROAD-4582)
- Upgrade Prometheus to v.2.45.0 (ROAD-4577)
- Upgrade Dockerfile to Debian Bullseye-Slim and generate latest version 20.x.x for the Asterisk container (ROAD-4581)
- Upgrade debian container distribution to latest GA (ROAD-4540)
- Remove legacy "kubernetes" folder from k8s-installer (ROAD-3639)
- Remove promLoki datasource (ROAD-4559)
- Remove alerts in Grafana based in v8 syntax (ROAD-4561)
- Remove wiface1 and nocache=true from wac.ini (ROAD-4585)
- Added query_retries and transaction_retries in Grafana configuration (ROAD-4556)
- Added timestamp to xmpp-server logs (ROAD-4696)
- Added timezone to Kitter deployment (ROAD-4764)
- Added flag "getCurrentInvitationsByUsername" to search current invitations by peer or username, defaults to false (cad265b)
- Creation of indexes in the QSS collections (ROAD-4316)
- Migrate Graph and Table to Time Series and Table in Grafana (ROAD-4164)
- Deprecate process-exporter (ROAD-4166)
- Move node-exporter to monitoring namespace (ROAD-4449)
- Fixed sip-mapping unit tests (ROAD-4645)
- Publishers will be blocked untils an alarm clears in the message broker (ROAD-4740)
- Change update DB method of rooms (ROAD-4914)
- Change internal_sample_rate parameter from 8k to "auto" for better audio quality in the audiomixer service (ROAD-4793)
- Search user by phone number now uses normalizePhoneNumber for consistency (cd7eb42)

## 3.9 v5.9.0

Released 17/07/2023

### 3.9.1 New features

- New “actionable alerts” graph in Grafana (ROAD-2385)
- New endpoint in kitter service (findUserByDomainandUsername) and new search option (“exact”) (ROAD-4428)
- New NOK answer in liveness and readyness QSS endpoints (ROAD-2715)
- Improve error log level when sending SMSs (ROAD-4523)
- Added a connection control log with RabbitMQ in the sfu-wrapper and recording liveness endpoints (ROAD-4326)
- Allow configuration of maximum file transfer size (ROAD-4504)

### 3.9.2 Bug fixes

- Avoid collection scans in the conference service to prevent slow querys by adding several new database indexes (ROAD-4350)
- Data is not removed in the dispatcher database in several cases, which could lead to a disconnection of the wrapper service (ROAD-4499)
- Do not retrieve all sessions in the database after an admin login in erebus service (ROAD-4528)
- Misconfiguration in the Kubernetes template of the alert section (ROAD-4189)
- ConferenceState service database is not correctly updated in some cases, which could lead to a call blockage for outgoing calls (ROAD-4493)
- Calltransfer service crashes when initiating a call in some corner cases (ROAD-4282)
- Recording postprocessing not working for calls without audio stream (ROAD-4140)
- Recording postprocessing not working when videos are turned off and on again (ROAD-2900)
- Pushes with p12 certificate don’t work giving a “Error load PFX” message (ROAD-4409)
- Users created on the fly via a ADFS login do not have the proper DisplayName information (ROAD-4278)
- Incoming requests have no session attached (ROAD-4263)
- registerXmppPushToken service call failed “database operation failed insert” (ROAD-4219)
- Orphan group sip-mappings are not deleted (ROAD-4156)
- Fallo en PUT en /usersGroup/<groupId> al cambiar los participants (ROAD-2806)
- DELETE /sapi/alerts/templates/{id} returns an unexpected response (ROAD-4297)
- Orphan sip-identity is not removed when a sip-mapping is deleted (ROAD-4249 )
- Concurrent conferences graph is incorrect in the monitoring dashboard (ROAD-4221)
- 404 error responses take too much time (ROAD-2657)
- The Janus-Wrapper and QSS loses connection with Asterisk’s ami (ROAD-4503)
- Missing meeting participants after restarting SFU (ROAD-3426)
- XMPP container does not persist messages after a reboot (ROAD-4398)
- The user is created again if it is deleted while logged in (VOD-182)
- Duplicated usernames and owners in SIP-mapping collection (ROAD-4218 )
- Account already exists error when doing ADFS log in (VOD-198)

### 3.9.3 Changes and deprecations

- Decouple validation tasks from callKSGW service (ROAD-4347)
- Deprecated Spohit backend (ROAD-4508)
- Update grafana version to 9.3.6 due to a security issue (ROAD-3590)
- Avoid collection scans (COLLSCAN) in the alert service to prevent slow querys (ROAD-4165)
- Add rtp_timeout and rtp_timeout_hold in the audiomixer configuration (ROAD-4328)
- Remove vExpress to connect to wac-core in the erebus service (ROAD-4150)
- Remove @quobis/vexpress from wac-core
- Messagewaitingindicator and sipidentitiespool services are now deactivated by default (ROAD-4452)
- Add nodeSelector disktype=ingress in the ingress-nginx configuration (ROAD-4431)
- Upgrade internal libraries, clean dependencies, unitiliaze rabbitmq related connections when service is stopped (ROAD-4476)
- Increase performance of get meetings of user use-case (ROAD-4355)
- Increase performance of retrieve groups where a user is participating usecase (ROAD-4352)
- Unprotect wac-core endpoint /meetings/byRoomId needs (ROAD-4397)
- Upgrade janus version from janus 0.9.2 to 0.12.4 (ROAD-4460)

## 3.10 v5.8.0

Released 28/04/2023

### 3.10.1 New features

- Support to Kubernetes cluster version 1.21 (ROAD-3635)
- Added audiomixers metrics to SIP monitoring dashboard (ROAD-3564)
- Reimplementation of "Current active sessions" chart using "wac-core_current_active_sessions_gauge" value (ROAD-2728)
- Reimplementation of "Logged users" chart using "waccore_current_logged_users_gauge" value (ROAD-2729)
- Reimplementation of "Active conferences" chart using "waccore_active_conferences_gauge" value (ROAD-2730)
- Support for Longhorn storage (ROAD-3640)
- Support for X.509 certificates (ROAD-3670)
- Support to XEP-0363 XMPP extension for filetransfer (ROAD-3683)
- Support for PostgreSQL connection from wac-core service (ROAD-3216)
- Created two different Keycloak client-id's for Android and iOS connections, namely collaborator-android and collaborator-ios (ROAD-3396)
- Added an endpoint to retrieve a meeting given its associated room ID (ROAD-3349)
- Added Grafana dashboards with lua metrics of the XMPP server memory consumption (ROAD-4157)
- Added new configuration for Prosody 0.12 and endpoint in Prometheus (ROAD-3663)

- Added Jaeger support for end-to-end distributed tracing (ROAD-2089)
- Customization of the burst settings for the websocket message rate (ROAD-3525)
- Update k8s-installer documentation about MetalLb configuration (ROAD-3638)
- Created username-domain index in collection "user" (ROAD-4178)
- Optimization of case-insensitive regex queries using username and email properties (ROAD-2394)

### 3.10.2 Bug fixes

- Fixed a memory leak in qss.io-websockets service (ROAD-2872)
- Deployments don't have a revision limit (ROAD-3243)
- Several errors in the use of Grafana with new volumes (ROAD-4203)
- Destruction of PVC in some services when performing destroy/redeploy (ROAD-4199)
- Prometheus and Loki has invalid claimName (ROAD-4204)
- Document and change the rpsHigh variable to rpsFront (ROAD-4105)
- Port 5070 is hardcoded in the SIP proxy template (ROAD-3495)
- Kitter logs shows internal opentelemetry traces (ROAD-3532)
- Username/email migration should be more tolerant when migrating incomplete documents (ROAD-4251)
- Cannot login as default admin when auth.userIdentifier is configured as *email* and email is not lowercase at core/defaults.js (ROAD-4207)
- Wrong database index in the "missedCalls" collection (ROAD-3729)
- Some Kubernetes resources cannot be recovered after a fresh installation (ROAD-4253)
- PostgreSQL database for wac-core is hard-coded (ROAD-4252)
- Ansible tag passwords-deploy does not generate "wacDatabasePassword" (ROAD-4233)
- Ansible tag cluster-destroy fails because some packages cannot be removed (ROAD-4232)
- Keycloak deployment process fails when x509 flag is not enabled (ROAD-4231)
- Missing providers folder for wac-core (ROAD-4230)
- Missing deployments folder for mongodb-exporter (ROAD-4228)
- Missing matchSelector in efs-provisioner deployment (ROAD-4273)
- Remove initContainer in wac-core (ROAD-4275)

### 3.10.3 Changes and deprecations

- The sipMappings data is now stored in a PostgreSQL database instead of MongoDB (ROAD-3216)
- Emails are now always stores lowercase (ROAD-2398)
- Convert automatically username and emails used as input values to lowercase (ROAD-2396)

## 3.11 v5.7.8

Released 26/09/2023

### 3.11.1 Bug fixes

- Fixed an issue that leaded to a wrong behaviour when calling to RingingGroups (this bug was introduced in 5.7.7)

## 3.12 v5.7.7

Released 23/09/2023

### 3.12.1 Bug fixes

- Fixed an issue that leaded to a "GetCurrentInvitations" message without content that affected clients using Callkit

## 3.13 v5.7.6

Released 18/09/2023

### 3.13.1 Bug fixes

- Wrong response code "603" instead of "200" in accepted calls to SIP destinations (ROAD-4775)
- Orphan group sip-mappings are not deleted (ROAD-4156)
- registerXmppPushToken service call failed "database operation failed insert" (ROAD-4219)
- Avoid collection scans (COLLSCAN) in the rooms service to prevent slow querys (ROAD-4366)
- Caller does not get a busy message when calling to a ringing group and the last candidate has several sessions and rejects the call (ROAD-4635))
- All users ared loaded in wac-core memory after deleting one of them, leading to a CPU peak (ROAD-4636)

## 3.14 v5.7.5

Released 17/07/2023

### 3.14.1 Bug fixes

- Wrong behaviour when calling to a ringing group and the "allowInvitesWhileInCall" parameter is set to false (ROAD)

## 3.15 v5.7.2

Released 28/06/2023

### 3.15.1 Bug fixes

- Clean-up database entry after a call is made (ROAD-3343)
- New Kitter endpoint (findUserByDomainandUsername) and new "exact" search method (ROAD-4428)
- Migrate existing emails and usernames to lowercase (ROAD-2395)
- Convert automatically username and emails used as input values to lowercase (ROAD-2396)
- Optimize current case-insensitive queries using username and email properties (ROAD-2397)
- Save emails in lowercase (ROAD-2398)
- Stop using CI queries to V2 users API in kitter (ROAD-4168)
- Fixed a memory leak in qss-io-websockets (ROAD-2872)
- Wrong database index in the "missedCalls" collection
- Cannot login as default admin when auth.userIdentifier is configured as *email* and email is not lowercase at core/defaults.js (ROAD-4207)
- The graph that shows the number of active conferences is incorrect (ROAD-4221)
- Username/email migration should be more tolerant when migrating incomplete documents (ROAD-4251)
- Users created on-the-fly via a third party ADFS do not stoep the "Displayname" in the presence collection (ROAD-4278)
- Calltransfer service crash when initiating a call in some circumstances (ROAD-4282)
- Pushes with p12 certificate don't work and fails with an "Error load PFX" error message (ROAD-4409)
- Rooms are not removed from the dispatcher service which leads to a disconnection from the SFU wrapper (ROAD-4499)

## 3.16 v5.7.0

Released 13/02/2023

### 3.16.1 New features

- Added a new Keycloak authenticator that allows to skip the step of selecting an identity provider. A user can now enter an email address and the system will then choose an identity provider based on the provided email domain and forward the user to the chosen provider (thanks to keycloak-home-idp-discovery plugin). Closes ROAD-3138.
- New ringing group strategy "simultaneous reduced" that complements the existing "simultaneous" and "sequential ones" (ROAD-3442)
- Added new Opentelemetry metrics from wac-core that exposes the following metrics, computed in a 20 seconds interval. This service needs to be activated in the wac.ini [opentelemetry] section (ROAD-3572):
  - waccore_active_conferences_gauge
  - waccore_current_active_sessions_gauge
  - waccore_current_logged_users_gauge
  - waccore_current_users_with_1by1_chats_gauge
  - waccore_total_users_gauge
- Added websocket throttle environment variables to erebus deployment (WAPI_WS_THROTTLE_BURST_COUNT, defaults to 2000, and WAPI_WS_THROTTLE_MS, defaults to 200) (ROAD-3575)
- Added currentTimeZone as a variable in the meeting templates (ROAD-3557)

### 3.16.2 Bug fixes

- Remove old deployments in the K8S installer for the QSS service (ROAD-3567)
- Port 5060 is hardcoded in the kamailio service template (ROAD-3496)
- `PSQL_Pass` variable has a wrong value in the maintainer's job.yml (ROAD-3556)
- The information of the presence parameter note is not saved (ROAD-3648)
- Erebus services loses connection with the message broker in some circumstances (ROAD-3611)
- wac-core loses connection with the message broker in some circumstances (ROAD-3610)
- Keycloak unusual CPU usage (VOD-187)
- Fix some typos in the WS documentation for qss/AudioStatus service (ROAD-3642 )
- No push notification when a user is added to a chat group and the mobile application is in background (ROAD-3068)

### 3.16.3 Changes and deprecations

- Avatars are now saved in the presence collection as a URL (`avatarUrl`) instead of a BASE64 blob . There are three new parameters to take into account in the wac.ini file: avatarStorageUrl, avatarStorageDir and avatarBase64Persistence (read the wac.ini template for more details). Added also a `bin/waccli presence create-avatar` script to migrate existing avatars from base64 format to URL format. (ROAD-2868)
- As result of the above change, from now on, Filesharing service is required by the resence service.
- Document the QSS WebSocket API (ROAD-3418)
- Authentication: do not request signing keys from keycloak for each authenticated request (ROAD-3603)
- Update xmpp-server version to 0.12 for better performance (ROAD-3484)

- New maintainer version 1.3.7 that adds two new values: DELETE_PARALLEL_SIP (defaults to false) and DELETE_TRUNK (defaults to false). This parameter will force a periodic purge of the "parallelSIP" and "trunk" collections in the database in case it's needed (ROAD-3474)

## 3.17 v5.6.0

Released 13/02/2023

### 3.17.1 New features

- A new QSS service ("qss-audiostatus") can now send mute/unmute requests to remote participants and also track user mute/unmute status (ROAD-2327 and ROAD-2287)
- The log verbosity level can now be configured on a per-service basics instead of a platform-wide value (ROAD-3240)

### 3.17.2 Bug fixes

- Dashboards shows an error "Failed to upgrade legacy queries Datasource $DS_Prometheus" (ROAD-3353)
- The configmap grafana-dashboards is not created because it is larger than 1MB (ROAD-3351)
- Remove spaces and reduce JSON names (ROAD-3352)
- Unconsistent naming on audiomixer-io objects (ROAD-3385)
- Addressbook variables "true/false" are set to uppercase causing wac-core to ignore them (ROAD-3361)
- Ansible installer broken when the inventory or vault-id contains a regular expression with "_" (ROAD-3354)
- A "keycloak-update" does a redeploy instead of updating the deployment (ROAD-3419)
- The SMS backend is always required in the wac-core template ROAD-3357
- The chat_instant_video variable does not reference to anything in the conf.js template (ROAD-3356)

### 3.17.3 Known-issues

- The audio status of the remote participants is not updated under some circumstances (ROAD-3487)

### 3.17.4 Changes and deprecations

- Updated SIP proxy version to 5.5.5
- Updated audiomixer version to 18.15.1 (ROAD-3250)
- Updated message-broker version to 3.11.7 (ROAD-3386)
- Updated SFU version to v0.12.4 (ROAD-3389)
- Audiomixer parameter dsp_drop_silence has been set to "no" for better performance.

## 3.18 v5.5.1

Released 13/02/2023

### 3.18.1 Bug fixes

- **Added two configuration variable to fine-tune the "erebus" websocket traffic throttle:**
  - `WAPI_WS_THROTTLE_BURST_COUNT` with default value of 100, consider that the SDK may send as much as 1msg/contact at startup in a very short timeframe
  - `WAPI_WS_THROTTLE_MS` with default value of 10, increase this value to wait longer to enter burst mode.

## 3.19 v5.5.0

Released 12/01/2023

### 3.19.1 New features

- Meetings can now be recorded. The recording can be activated or deactivated by the user on a per meeting basis. An overlay text banner can also be added to the recording, whose content is also configurable by the administrator. A new REST API endpoint is also available to retrieve information about these recordings and its participants (ROAD-1026, ROAD-1839,ROAD-2707).
- Added a new endpoint `phonebook/all` that allows to retrieve the phonebooks whose name matches a given filter (at least 4 characters are required to filter (ROAD-2880)
- Added a new method to edit an existing phonebook entry, `patch /phonebook/{phonebookId}/contact/{contactId}`) (ROAD-2853)
- In order to prevent DoS attacks, a new throttle mechamism has been implemented in the incoming wac-core websocket connection If the rate limit of 10 messages per milisecond is exceeded on a given connection, the system enters in a burst mode where a total of 100 messages are allowed. Otherwise, the connection is dropped with a 4429 status code and this log is provided: "Client reached maximum messages allowed in burst mode. Disconnecting!". This count applies to each individual WS connection and counts only the messages in the inbound direction (client to server). (ROAD-3341)
- A maximum and minimum default resources value can now be set for each pod. That will help to prevent system overloads (ROAD-2684).
- Add tags and corresponding variables for metrics-server and HPA server (ROAD-2683)
- Added new graphs to check the size of relevant database collections in Grafana (ROAD-2815)
- Grafana version updated to v9 (ROAD-2809)
- Added several microservices variables to the Ansible installer (ROAD-2688)
- Changed wrapper service log RoomAccessLog message, replacing "join: false/true" into "event: join/leave", which provides more information (ROAD-3149)
- Added a log message to the "get presence" endpoint with the requested presence address (ca99d4e8).
- Added warning-level log messages when the XMPP client tries to address a MUC and its XMPP address is empty, ie, the group has no *chatId* assigned: "Seems that I have no admin XMPP client available, refusing to try to send an invite", "Will try to invite an empty group XMPP address!" and "Will try to revoke membership from an empty group XMPP address!" (4caa4492)

### 3.19.2 Bug fixes

- Crash loop back error when deploying Loki 2.4.2 (ROAD-3227)
- Typo in the sippo-maintainer job template that was causing a malfunction in this service (ROAD-3226)
- The value "turn_secret" was not included into the vault (ROAD-3169)
- The PostgreSQL configMap "postgresql-conf" is not removed when performing the "postgresql-destroy" Ansible task (ROAD-3166)
- A phonebook entry is stored twice when the same contact is set (ROAD-3168)
- The user is not notified that they have a missed call when the user is offline in parallel-sip configuration (VOD-185)
- Optimization defaults too high for PostgreSQL deployment (ROAD-3167)
- SIP mapping endpint answers with a wrong header when the SIP identities pool service is active (ROAD-3015)
- Wrong error code sent when revoking a session in wac-core after a new one is established. (ROAD-2863)
- Corrected some push errors that are not correctly printed in the logs (7c8f95c2)

### 3.19.3 Changes and deprecations

- wac-core:
  - login.toml's exclusive option has been removed in favour of session.toml's `singleUserSingleSession` one.
  - A new service configuration is required in wac-core in order to activate meeting recording ("[recordings]"). In addition, a new shared volume is required (this is alreaday taken into account by the installer).
  - "On hold" requests via /sapi/kmanage/siponhold are now deprecated (ROAD-3112).
  - Adapted WAPI/Erebus and @quobis/peers to allow propagating the close reason set by wac-core downstream. In the event of a server-side session close, the reason code will be 4401. An SDK implementor should treat that 4401 code as a logout, so the local status of the session must be unauthenticated.
  - REST API documentation is now alphabetically ordered.
- Recording:
  - a new parameter `customize_user_video_tag_text` added in recording.toml to indicate the content of the overlay text. Available variables are {{username}}, {{domain}} and {{id}}. You can combine them and use static texts, for instance `User: {{username}}@{{domain}}`
  - removed log "Recordings folder: that.config.server.recording" that was misleading (ROAD-2762)
- PostgreSQL service: values changed as follows for settings `shared_buffers` and `work_mem`:
  - default values: shared_buffers = 128MB, work_mem = 4MB
  - recommendation for 4GB RAM: shared_buffers = 750MB, work_mem = 10MB
  - recommendation for 8GB RAM: shared_buffers = 1,5 GB, work_mem = 20MB

### 3.19.4 Known issues

- Kubernetes installer issues:
  - Dashboards "Main Dashboard" and "MongoDB" does not load becausea of misconfigured installation config (ROAD-3353)
  - The chat_instant_video variable does not reference any in the conf.js template (ROAD-3356)
  - The SMS backend needs to be configured (setting backend_sms_enabled), otherwise the wac-core service does not start (ROAD-3357)

## 3.20 v5.4.0

Released 22/11/2022

### 3.20.1 New features

- Monitoring: new dashboard for Kitter service
- Monitoring: Prometheus nodePort exposed
- Maintainer: added metrics-server deployment v0.5.1
- Added a performance configuration for PostgreSQL database
- Added logs to wrapper service about the status of the SFU connection (log: "Dispatcher client error/reconnecting/disconnected!") (ROAD-2156)
- File user.toml is no longer required to start the wac-core service. A log is provided to notify that default values will be used (ROAD-2640)
- Added new logs about issues with the push notification services, details *here* (ROAD-2668)
- Assign timezone and locale to anonymous participants (same as meeting owner) (wac-core 7357088c)

### 3.20.2 Bug fixes

- Config: fix keycloak connection URL
- Monitoring: fix broken Grafana input fields
- Deactivating missedCalls service blocks conferenceLog writing (ROAD-2658)
- REST-API answers with 500 error code when a path doesn't exist (ROAD-2653)
- Wrong meeting PUT event payload ("isPasswordProtected" instead of "password") (ROAD-2649)
- 400 Bad Request returned when trying to remove password protection from A meeting (ROAD-2648)
- Existing quick-conferences allow only three participants after a system reboot (error log: maximum number of publishers (3) already reached) (ROOM-346)
- The chats of deleted users remain indefinitely in the list of chats (VOD-146)
- The /auth/realms/master/metrics endpoint is now working with latest Keycloack version (ROAD-2674)
- The keycloak deployment misses the parameter nodeSelector: disktype (ROAD-2667)
- Amend translation of SS7 Q850->http error codes (the mapping of some codes to HTTP where not accurate) in the QSS

- Do not send notifications to users when a meeting that is due is deleted (only to the meeting owner) (ROAD-3150)
- Do not require to provide a password when updating a password-protected meeting. This was impeding the removal of a password for a meeting (wac-core f8bdf82e)

### 3.20.3 Changes

- Setting a value of *0* to the notification expiration header will instruct APNs to not retry at all sending the notification to the device (push-clients d1c81334)
- Changed some error messages levels in the erebus service. Example: "Attaching session session.id and user session.user to connection connection.id" is now "info" instead of "debug".

## 3.21 v5.3.1

Released 26/09/2022

### 3.21.1 New features

- Allow to setup the user group ringing strategy and its parameters via REST API (ROAD-2390)
- Persistent and centralized missed call counter (ROAD-2016).
- Added more specific log-level logs for Erebus service in info & error levels (VOD-110)
- Include email and phone number values for domain contacts in the AddressBook's endpoint response body (QUOBDEV-1753)
- Upgrade Keycloak component to latest upstream release (ROAD-2259)
- Document "401 error" in the REST API documentation(ROAD-2546)
- Make the homoglyph check configurable (ROAD-2456)

### 3.21.2 Bug fixes

- Meetings can be updated by regular users without admin priviledges under certain circumstances (SEC-167)
- Under some circumstances, push notification reaches the other participants in a ringing group after one of them answers the call (ITX-647)
- Database migration 26.5 and 26.6 fails (ITX-653)
- Blocked non authorized access to Keycloack API (ROAD-2567)
- Fixed single-session support (ROAD-2311)
- A malformed *room-destroyed* event is sent in a ringing group call (ROAD-2521)
- A "PUT" query to the "domain" endpoint accepts an invalid parent (ROAD-2449)
- Meeting owner data is not updated when meeting details are updated (ROAD-2441)

### 3.21.3 Known issues

- Deactivating missedCalls service blocks conferenceLog writing. Workaround: always initialize the missedCalls service. (ROAD-2658)
- Meetings cab be updated by another user different from the owner (SEC-167)
- A push notification arrives to the mobile applications if a call has been made while the device was out of connection (ITX-661)
- 400 Bad Request returned when trying to remove password protection from A meeting (ROAD-2648)
- The /auth/realms/master/metrics endpoint is now working with latest Keycloack version (ROAD-2674)
- The keycloak deployment misses the parameter nodeSelector: disktype (ROAD-2667)

### 3.21.4 Changes

- New "missed call" services that requires specific activation at the wac.ini configuration file in case of migrations of previous systems ([missedcalls]). This service is activated by default in new installations.
- Two new parameters for setting the default local and timezone (defaultTimezone and defaultLocale) in file wac/config/user.toml
- Do not delete presences when deleting users, as we asume all users have a presence so if we drop presence for deleted users, applications querying data about those users may fail. Instead, when a presence query about a deleted user is done, the answer is an empty presence (it only contains the address field, the other fields are null or empty strings)(wac-core 20b0ca3b)

## 3.22 v5.2.2

Released 22/08/2022

### 3.22.1 Bug fixes

- Calls from PSTN users to internal users are not registered in the conference log under some specific situations (VOD-160)

## 3.23 v5.2.1

Released 01/08/2022

### 3.23.1 Bug fixes

- Unanswered calls not shown in the callee conference log in parallel-SIP environments (VOD-97)
- Calls to offline users do not appear in the call-history of the callee (VOD-98)

## 3.24 v5.2.0

Released 15/07/2022

### 3.24.1 New features

- The ringing group strategy configuration, which was introduced in version 5.1.0 as system-wide configuration, has been improved in this version and it's now a per-usergroup configuration. This is implemented in a new object named "ringingstrategy" available in the usersGroup endpoint of the wac-core REST API (ROAD-2468).
- There are now two new parameters to setup the user local timezone and its locale ("currentTimezone" and "locale"). These two new properties are used by the wac-core service when sending SMS and meetings notifications and also in the conference log generation (ROAD-2459).
- New parameter "chatId" added to the UsersGroup model that includes the ID of the associated chat room (QUOBDEV-1900).
- Added a new library @quobis/opentelemetry-wrapper (ROAD-2330).

### 3.24.2 Bug fixes

- Corrected wrong behaviour that caused the wrapper service to reboot with a log message "TypeError: this.deleteRoom is not a function" (VOD-107).
- QSS-meeting-basic service could not start if service qss-registry was not available (ROAD-2247).
- wac-core service allows case-sensitive "groupID"s while the XMPP services is case-insensitive. A check has been added to prevent this problem in the future (ROAD-2410).
- User update API in WAC-core allows to modify values incongruously. Now, The fields "created" and "lastLogin" have been ignored from the PUT request - users. Therefore, even if any of these fields comes in the body, no update will be done (ROAD-2354).

### 3.24.3 Changes and deprecations

- Ringing group strategy configuration from config.json has been deprecated and removed. This information is now avaible at each ringing group.
- Migration of wac-core 26.4 is required (wac-core: node bin/waccli migrate -f 26.4).

## 3.25 v5.1.2

Released 05/07/2022

### 3.25.1 Bug fixes

- Incoming calls to user groups participants when the mobile app was running in background mode were not automatically cancelled in sequential ringing (ROAD-2434)

## 3.26 v5.1.1

Released 21th June 2022

### 3.26.1 Bug fixes

- Calls to user groups with sequential strategy are now working properly in mobile devices (ROAD-2293)
- Changed wrong internal reference in erebus service from "wac" to "wac-core" (QUOBDEV-1937)

## 3.27 v5.1.0

Released 06/06/2022

### 3.27.1 New features

- Support for push notifications when the wac-core service sits behind a HTTP proxy (VOD-136)
- New dashboard that provides login information from Keycloak such as logins/logouts per second, per domain, login response time, login errors, etc... (ROAD-2011)
- New dashboard with metric of the Kitter service (ROAD-2086)
- New *ringing strategy* for UserGroups: sequential ringing. This is a system-wide configuration option via two new parameters: "stepDelay" and "maxSteps" (ROAD-2154)
- A new endpoint "putParticipants" with path `/sapi/usersGroup/:groupId/participants` has been added to the wac-core REST API in order to change/update the participants of an userGroup and it's order (ROAD-2168)
- Kitter API can now handle users without sipmapping attribute (ROAD-2218)
- Kitter API can now handle users without wacUserId attribute (ROAD-2219)

### 3.27.2 Bug fixes

- All established calls are automatically terminated when the "stepDelay" * "maxSteps" time is reached and sequential ringing is activated (ROAD-2229)
- User created on the fly can now be deleted using Kitter API (ROAD-2204)
- User matching in login process is now case-insensitive (ROAD-2225)

### 3.27.3 Changes

- There are three new enviroment vars for Kitter configuration: JAEGER_HOST (host where is the jaeger endpoint for monitoring traces), JAEGER_PORT (port of the the jaeger endpoint for monitoring traces) and PROMETHEUS_PORT (port of the prometheus endpoint to send metrics to).

### 3.27.4 Known issues

- The chats of the deleted users remain indefinitely in the chat list (VOD-146)
- The meeting time shown in the email body is different from the time shown in the ICS file (VOD-146)
- Single session: inability to properly logout earlier session because of keycloak's cookie (ROAD-2311)
- Keycloak shows an error on first login (and creates the user) when login using ADFS (ROAD-2191)
- Logout is not working when authenticating against ADFS server (ROAD-2190)
- Changes in unique key `mobilephone` were not properly managed on system boot (ROAD-2160)
- Calls to user groups with sequential strategy are now working properly in mobile devices (ROAD-2293)
- Unanswered calls are not shown in the callee conference log with Parallel-SIP (VOD-97)
- On multiparty calls, when a participant hangs, he is still shown for the rest of participants (QA-2941)
- The list of chats does not show the conversations before the version update (QA-2907)
- Instant audios sent by iOS native app to a web user are not received (QA-2790)
- Finishing a call as callee on PSTN does not finish it on the platform (QA-2578)
- The call of a multi-device users ends automatically when users log out from another device (QA-2513)

## 3.28 v5.0.5

Released 24/05/2022

### 3.28.1 Bug fixes

- Wrong timestamp of the recording files (ROAD-2189)
- Empty mobile phone field in the user on-the-fly creation caused a wac-core crash (ROAD-2183)

### 3.28.2 Known issues

- Unable to delete user created on-the-fly via ADFS using Kitter API (ROAD-2204)
- Keycloak shows an error on first login (and creates the user) when login using ADFS (ROAD-2191)
- Logout is not working when authenticating against ADFS server (ROAD-2190)
- Changes in unique key `mobilephone` were not properly managed on system boot (ROAD-2160)
- Unanswered calls are not shown in the callee conference log with Parallel-SIP (VOD-97)
- Information button doesn't display call details when screen-sharing functionality is activated (QA-2996)
- On multiparty calls, when a participant hangs, he is shown for the rest of participants (QA-2941)
- The list of chats does not show the conversations before the version update (QA-2907)
- Instant audios sent by iOS native app to a web user are not received (QA-2790)
- Finishing a call as callee on PSTN does not finish it on the platform (QA-2578)

- The call of a multi-device users ends automatically when users log out of the device where they do not have an established call (QA-2513)

## 3.29 v5.0.4

Released 12th May 2022

### 3.29.1 Bug fixes

- Ansible deployment can be done now with Debian 11 (DEVOPS-501)
- Session restore was sending a `404 Bad Request` under specific circumstances. That means that mobile apps couldn't recover from background state (ROAD-2157)
- Single-session parameter was not working due to a problem with Keycloack sessions (ROAD-2158)
- No response is sent to client when an already created ADFS user was recreated (ROAD-2149)
- A malformed AMQP message caused the wac-core service to crash. Now, the malformed event is handled correctly and a log message is saved ("Unable to parse event, ignoring. The event content was: ...") (AS-141)
- Login using a remote ADFS is not working (DEV-1811).
- The `id@anonymous` string appears when the user accesses the lobby of a meeting (VOD-114).
- VoIP pushes has a 1-hour lenght timeout, fixed now to 30 seconds (DEV-1925).
- Cannot sort users by email via the REST API (ROAD-2083).
- SMS templates of meetings has name and URL as mandatory fields (ROAD-2095).
- Under some circumstances the incoming request has no socket assigned (ROAD-2128).
- Images with webp format are rejected for user's avatar (QA-2985).

### 3.29.2 Known issues

- Unable to delete user created on-the-fly via ADFS using Kitter API (ROAD-2204)
- Keycloak shows an error on first login (and creates the user) when login using ADFS (ROAD-2191)
- Logout is not working when authenticating against ADFS server (ROAD-2190)
- Wrong timestamp of the recording files (ROAD-2189)
- Empty mobile phone field in the user on-the-fly creation caused a wac-core crash (ROAD-2183)
- Changes in unique key `mobilephone` were not properly managed on system boot (ROAD-2160)
- Unanswered calls are not shown in the callee conference log with Parallel-SIP (VOD-97)
- Information button doesn't display call details when screen-sharing functionality is activated (QA-2996)
- On multiparty calls, when a participant hangs, he is shown for the rest of participants (QA-2941)
- The list of chats does not show the conversations before the version update (QA-2907)
- Instant audios sent by iOS native app to a web user are not received (QA-2790)
- Finishing a call as callee on PSTN does not finish it on the platform (QA-2578)
- The call of a multi-device users ends automatically when users log out of the device where they do not have an established call (QA-2513)

- Existing quick-conferences allow only three participants after a system reboot (error log: maximum number of publishers (3) already reached) (ROOM-346)

## 3.30 v5.0.0

Released 25/03/2022

**Backward compatibility announcement**

Please take into account these limitations about backward compatibility following the announcement of Quobis Communication Platform v5.0:

- Quobis native application (SDKs and collaborator) v5.0 are fully compatible with systems running Quobis wac v5.0, v4.4, v4.3.
- Quobis Javascript SDK v5.0, Quobis collaborator web & desktop v5.0 are not compatible with systems running Quobis wac v4.4 or v4.3.
- Quobis SDKs v4.x , Quobis collaborator v4.x are not compatible with systems running a Quobis wac 5.0.
- Quobis Manager v5.0 is not compatible with systems running Quobis wac 4.x.
- Quobis Manager v4.x is not compatible with systems running Quobis wac 5.0.

Version 4.4 of Quobis Communication Platform products (Quobis wac, Quobis SDK, Quobis Manager) will be supported until November 2022. Please take this end-of-life date into consideration as per the implications stated above.

Released 25th March 2022

### 3.30.1 New features

- User login and authentication process has changed in v5.0 in order to provide a better and more secure authentication process. Two new elements have been added in v5.0: Keycloack (a third-party open source IAM server) and Kitter (a service responsible for user provisioning and coordination between Keycloack and wac service). User authentication and provisioning is made now using these new services and their associated APIs, instead of using the wac REST API endpoints. Please note that this new implementation **breaks the REST API contract** as explained in the backward compatibility announcement above.
- Configure the system to be able to connect an external debugger using –inspect parameter (QUOBDEV-1307)
- Improved connection error management related to AMQP messages (QUOBDEV-1670)
- REST API new features and improvements:
  - New REST API conferences endpoint with cursor based pagination (QUOBDEV-1422)
  - Allow regular users to query other users via REST API (QUOBDEV-1565)
  - Enhance documents information with username and domain in the presence REST endpoing (QUOBDEV-1564)
  - Improved users REST API endpoint to get users by their username and name
  - Create sipmappings V2 endpoint (QUOBDEV-1402)
  - Add new pagination and filters to sipmappings endpoint (QUOBDEV1349)
  - Create users groups V2 endpoint (QUOBDEV-1403)
  - Add partial searchers in Users string fields (QUOBDEV-1348)
  - Add new pagination to users endpoint (QUOBDEV-1343)

  - Add new pagination to group endpoint (QUOBDEV-1353)
  - Return the total number of records per collection in the pagination library (QUOBDEV-1341)

### 3.30.2 Changes

- Service "sippo-server" has been renamed to "wac-core".
- Deployment software package "sippo-k8s" has been renamed to "k8s-installer"
- There is now a domain named "anonymous" that is automatically created by default and that cannot be edited or removed. All anonymous users are mapped into this domain, which is a "child" of the main "quobis" domain.

### 3.30.3 Bug fixes

- Erebus limits the number of connection to the QSS from a given endpoint to prevent system overload, with the exception of WS *janus-protocol* as there can be more than one connection for call transfers, call-hold, etc... A log has been also added ("Cannot open a new pipe for the same server ${server} and protocol ${protocol} from client ${clientId}").
- Presence is not being updated for some users (QA-2652)
- Watchdog registry checks are executed every millisecond (QA-2840)
- Erebus reached inconsistent state after a Kubernetes node was down
- Memory leak in qss-registry service
- Erebus fails to reboot after a node crash
- Static contact from Addressbooks is received without a phone value (QA-2849)
- First chat messages sent to a user are not received (QA-2850)
- An iptables rule needs to be added every time a kube-proxy restarts
- The xmpp-server-nginx does not connect to the sippo-server when it changes its IP address (QA-2051)
- The xmpp-server doesn't connect to postgresql after an environment crash (QA-1863)
- The response codes are different depending if the callee is a pstn or a domain user (QA-1754)
- Fix timeout on error when sending a SMS after deletion of a meeting (QA-2788)
- Reconnections and pod-restart issues:
  - Erebus fails to reboot after a node crash
  - Wrapper and recording services do not automatically reconnect after a RabbitMQ crash longer than three minutes (QUOBDEV-1679)
  - The dispatcher does not work correctly after a reboot
  - Error "restify.NotFoundError is not a constructor" appears in dispatcher logs
  - qss-trunk service needs to be the last QSS service to be initialized, otherwise it will refuse to start
  - qss-trunk service stops working properly and prevents outgoing calls in some circumstances
- REST API fixes:
  - Error 404 is returned when PATCH is sent to /api/user/ with phone_number and sip_mapping
  - 200 OK is returned when POST/PATCH to /api/users or /api/domains is sent with an invalid body

  - User REST API allows to change the user domain (QA-2471)
  - User REST API allows to update username (QA-2816)
  - Users REST API allows to update an alias using an empty value (QA-2812)
  - Clean orphan sip-mappings without an existing owner or user group (QA-2365)
  - Regular users can edit the data of other users through user SAPI under some circumstanace (QA-2820)
  - User search based on email with special characters is not working (QA-2698)
  - System allows to update an user using invalid user role value (QA-2469)
  - Users can have its role changed to anonymous even if their domain doesn't allow it (QA-1760)

### 3.30.4 Known issues

- User can not be changed from one domain to another
- On multiparty calls, when a participant hangs up, he is still shown for the rest of participants (QA-2941)
- Invites to PSTN: fromDetail.displayName is wrong, it displays the caller's number instead of the SIPs displayName (QA-2746)
- The call ends when canceling an invite participant request sent as callee (QA-2530)
- The SipMapping is not removed from the database when the userGroup is deleted (QA-2376)
- Recording does not process when videos are turned off and back on (QA-2271)
- Meeting start and end time sent by email are incorrect
- Users removed from the system remain listed as members of the chat groups to which they belonged (QA-1942)
- Meeting PIN is not being send on the SMS (QA-1938)
- Active conferences has not an associated chat (QA-1921)
- The xmpp-server doesn't connect to postgresql after an environment crash (QA-1863)
- A malformed requests in erebus should respond with 400, not 500 (QA-1768)
- Allow WAC to locally query itself instead of using public URL when validating an access token (QA-1761)
- Credentials OpenAPI definition missing query parameter (QA-1762)
- There are some users registered with multiple Firebase push tokens (QA-1755)
- When a disconnected user's ID is called, the call is redirected to PSTN (QA-1757)

### 3.30.5 Deprecations

- Internal service oAuth-2P (O2P) has been deprecated in v5.0 as it's been replaced by Keycloack service.
- Oracle, Ericsson and Dialogic gateway support has been deprecated in v5.0

## 3.31 v4.4.1

Released 17/05/2022

### 3.31.1 Bug fixes

- Wrong timestamp of the recording files (ROAD-2189)
- Addressbook service is monitoring events from all other services, even when disabled by config (QA-2607)

## 3.32 v4.4.0

Released 17/09/2021

### 3.32.1 New features

- meetings: improve email and SMS notifications' configuration
- meetings: react to deleted users that participate in meetings
- meetings: add timezone information to meeting templates
- meetings: allow complete disabling of notifications
- xmpp: provide connection information in hosts-meta files
- sipmappings: assign SIP mapping on login from a pool
- conflog: timezone information support for participants
- billing: billing information service
- user: return 409 on conflict when updating/creating instead of 403
- wac: inform web app root URL via hosts-meta files
- presence: ensure no duplicate subscriptions are created
- db: do not recreate already-existing indexes on restart
- mwi: send *subscribe* requests to SIP proxy on session recover

### 3.32.2 Bug fixes

- users: fix updating mobile phone field
- mwi: fix handling of new sessions
- user: fix phone number normalization and remove duplicated country code
- resolver: fix to allow resolving users with * in their username

## 3.33 v4.3.4

### 3.33.1 Bug fixes

- meetings: stop parsing users meetings field

## 3.34 v4.3.3

Released 10/05/2021

### 3.34.1 Bug fixes

- sipmappings: assign sipMapping automatically in login
- sippools: change collection name to use md5 instead of plain domain

## 3.35 v4.3.2

Released 02/05/2021

No changes

## 3.36 v4.3.1

Released 28/05/2021

### 3.36.1 Bug fixes

- QSS `room-update` event is not being sent when there is a ACL modification (QA-2276)

## 3.37 v4.3.0

Released 14/05/2021

### 3.37.1 New features and changes

#### 3.37.1.1 Global

- A new *permissions* service that allows to setup policies per domain and user and consume them via the REST API.
- Added "screensharing" capability by default in the domain creation
- An event `speakerChanged` is generated when a participant in a conference starts or stops speaking. This event can be consumed from the SDKs to provide this information to the rest of participants. Available in the Javascript SDK.
- An event is generated when a participant starts/stops sharing his/her screen

#### 3.37.1.2 Authentication

- Added support for OpenID as an external authentication service

#### 3.37.1.3 Deployment

- New feature "Blue-Green deployment" that allows to have two enviroments in parallel. That means that it can be used to migrate users from existing environment to a new one in order to test that everything works properly. This migration can be done on a per user basis updating the info of the user by using the Service API.

#### 3.37.1.4 Meetings

- Ability to set and start time, end time and meeting password

#### 3.37.1.5 Recording

- New set of *recording permissions* that allow fine-grained control on who can record what
- Three new levels of quality recording
- Recordings can be provided merged or separately into single files.

#### 3.37.1.6 Telephony and PBX

- Added MWI ("Message Waiting Indicator") *integration* with SIP-based external voicemail platforms. In addition, the voicemail SIP address is now also reachable via a generic destination string ("voicemail").

### 3.37.2 Bug fixes

- Fixed a bug that prevented SIP users to leave the conference room under some circumstances (QSS-284)
- Unneeded Credentials added to STUN ICE servers in the Dispatcher (QA-1824)
- Conference log update events are sent as many times as a user appears as candidate in a call (QA-1923)
- Improve log line in QSS registry (FOEHN-591)
- DataPipe export "symbol" instead of the proper enum (QA-1864)
- Meeting PIN is not being send on the SMS (QA-1938)
- Users removed from the system remain listed as members of the chat groups to which they belonged (QA-1942)
- Active conferences has not an associated chat (QA-1921)
- All alerts are displayed as send/received at current time when alerts tab is opened (QA-1943)
- Push notifications (iOS): sound property must be set to *default* (QA-1854)
- MWI does not sent sipMapping password in unregister (QUOBDEV-1059)
- Wrapper restarts if the plugindata received is empty (QA-1749)
- QSS-watchdog-registry stop working if database connection is lost (QSS-276)
- An anonymous user can delete itself via API (SRV-1722)

- Anonymous users can modify their own data via API (SRV-1723)
- Alert iOS Pushes are not working properly (QUOBDEV-827)
- Calls from the native app do not connect (QA-1668)
- Push notifications of chat non-text messages are not received (QA-1658)
- Contacts marked as favourite loose its value when reloading the browser (QA-1615)
- Alert template parameters are set to "null" when updating any of their values (QA-1692)
- Contacts API returns a 404 error when a user tries to update a contact (QA-1573)

### 3.37.3 Known issues

- Users removed from the system remain listed as members of the chat groups to which they belonged (QA-1942)
- Meeting PIN is not being send on the SMS (QA-1938)
- Active conferences has not an associated chat (QA-1921)
- The xmpp-server doesn't connect to postgresql after an environment crash (QA-1863)
- A malformed requests in erebus should respond with 400, not 500 (QA-1768)
- Users can have its role changed to anonymous even if their domain doesn't allow it (QA-1760)
- Allow WAC to locally query itself instead of using public URL when validating an access token (QA-1761)
- Credentials OpenAPI definition missing query parameter (QA-1762)
- There are some users registered with multiple Firebase push tokens (QA-1755)
- When a disconnected user's ID is called, the call is redirected to PSTN (QA-1757)
- The response codes are different depending if the callee is a pstn or a domain user (QA-1754)

### 3.37.4 Technical improvements

- Updated restify to 8.5.1 to fix security vulnerabilities (QUOBDEV-824)
- Automatically fix npm security vulnerable packages (QUOBDEV-823)
- Drop apn dependency in the QSS (QUOBDEV-822)
- Make addressbook service compatible with new static contacts service (QUOBDEV-819)
- Deprecate old *dnie* service (ROAD-1328)
- Deprecate old click2call service (ROAD-1327)
- Deprecation of sippo-cob-static (ROAD-1326)
- Wac <> xmpp events must be sent in chronological order (QUOBDEV-769)
- Optimization of framerate for screensharing (SIPPOJS-739)
- The inviteMany method should has the same behaviour than the regular invite
- When groups are requested via the Addressbook service, the received id in the response is the groupId

## 3.38 v4.2.7

- Temporary recording files are not deleted when configured to record in "audio only" mode (QA-1681)

## 3.39 v4.2.5

- ConferenceLogs events were not being sent to the caller when the call was canceled (QA-1850)
- Missing "sound" property default value for iOS notifications (QA-1854)

## 3.40 v4.2.0

Released 08/01/2021

### 3.40.1 New features and changes

- A SIP load balancer service is now available to route incoming SIP calls to a pool of audiomixers using several call routing strategies (round robin, priority based, etc...) for better performance and scalability.
- Support to SIP authentication using INVITE & REFER authentication has been added so to the SIP proxy.
- A new service called "Erebus" has been included in the architecture. Erebus is an API gateway that implements a websocket multiplexer that enables clients to access all of the system services using just one single connection. It reduces login and setup time (less TLS handshakes) and reduces the required bandwidth. This new way of connecting to the server is backward compatible.
- Three client websocket connections (into wrapper, sippo-server, and QSS services) have been multiplexed via Erubus into a single websocket connection. This change reduces significantly the call setup and connection setup times. More connections will be passed via Erebus in the upcoming versions.
- Added a websocket keepalive message in the websocket connection from the clients in order to prevent the drop of the connection by a firewall. This ping message is automatically sent by the SDKs each 75 seconds and is answered by the system (wrapper service). The 75 seconds value is hardcoded and cannot be changed.
- Asterisks are allowed as part of the username and alias fields.

Support to several redirect URIs in the oAuth2 authentication process in order to support several type of devices (desktop browser, mobile browser, native app, etc...) with a single oAuth2 host.

- Added a new functionality "agent assigner" that allows the dispatching of incoming calls to a set of users depending on several criteria. The assignment decision can be taken by an external system (typically, a contact-center or PBX) or locally by the Quobis wac.
- A "404 Not Found" SIP error code is sent as a failure message when user does not exist in the system or if the domain in the Request-URI does not match any of the domains handled by the recipient of the request, as defined in RFC-3261.
- SDK clients can now register into a SIP or IMS network via a SIP REGISTER. A valid registration on the system will triger a delegated SIP registration into an external SIP network using the ims-credentials values. Please note that this configuration has been deployed in stand-alone configuration only.

- Conference URLs path for public rooms is now configurable and set by default to `htttps://wac-public-uri/c/public_room`. This change has been done to isolate meeting URL from conference URLs
- A major refactor has been made in order to provide a easiest way to manage and consume contact information. Some methods have been deprecated in the Javascript and iOS SDK. Two new services, "UserContacts" and "Addressbooks", have been created as result of this refactor.
- Support to "Lightweight Kubernetes K3s" deployments (www.k3s.io) in addition to standard Kubernets K8s deployment. K3s is a highly available, certified Kubernetes distribution designed resource-constrained systems. Quobis recommends k3s deployments for lab and testing deployments.
- Deployment using docker-compose is no longer supported.
- The audiomixer, SFU and SIP-proxy services are now running as pods inside the Kubernetes cluster.
- K8s cluster can check the health status of the XMPP, o2p, sippo-server, wrapper, dispatcher and QSS containers.
- A storage quota parameter (maxStorageQuota) can be set in the installation process via the Ansible installer. This parameter is optional, applies to the entire namespaced and defaults to 60GB for the disk space used by databases, uploaded files, chats and backups. It the parameter is removed, there is no space limitation.
- Ability to configure the system to use dynamic TURN credentials to prevent fraudulent usage of the media server and DoS attacks. This functionality is configured by setting two new parameters in file `quobis-dispatcher-config.js` ("`secret`" and "`expirationHours`")
- Meetings can be optionally password-protected for better privacy and security. This password protecion applies both for system users and external users as well.
- Meeting organizer automatically receives the meeting invitation email when the meeting is created.
- Meeting URLs path for public rooms is now configurable and set by default to `htttps://wac-public-uri/m/meeting_ID`. This change has been done to isolate meeting URL from conference URLs
- Added "`VALARM`" support into the attached ICS file of the meeting invitation emails as defined in RFC2445. That allows the calendar client to raise a reminder before the meeting starts. This time can be configured in the "`remindBefore`" parameters, which defaults to 10 minutes.
- Join/leave group announcements are now available in the chat history.
- Two different Callkit push certificates can be configured for the same APNs: one for VoIP pushes and another for chat pushes.
- Mobile push notification payload now includes contact information of the caller party, following the standard order (contact name, display name, username and finally phone number) so the applications don't need to resolve that information locally.
- Added parameter "recordingType" that allows the selection of which media streams are recorded: audio and video, only audio, only video or nothing at all.
- Audio and video raw files used during the preprocessing phase are deleted from the temporary folders after they are successfully processed if the configuration parameter "clean_temporary_files" in config/config.json is set to true
- Ability to set a client IP source ranges to be excluded from rate-limiting (configuration option: limit-whitelist). This is useful in some specific scenarios like contact-centers where most of the traffic is coming from the same IP range.
- A password policy can now be set with some rules such as minimum length, minimum number of letters/numbers, lowercase/uppercase usage, symbols and spaces, password blacklist, etc...) in order to comply with ISO-27001 requirements. These policies are system-wide and can be configured at file `config/passwordpolicies.json`
- Restricted avatar uploads to base64 format URI with images to prevent potential XSS attacks.

- Added a homoglyph check to reject usernames that might look alike to the existing one, in order to avoid homograph attacks and typosquatting. For example, that happens where the Latin character "a" (Unicode U+0061) is replaced with the Cyrillic character "" (Unicode U+0410)

### 3.40.2 Bug fixes

- System was not taking into account the participants limit value when we have users with different media options (JW-63)
- It was possible to create and domain with an empty domain name which leaded to a login failure (SRV-1692)
- System does not start when the agent assigner services is enable and there are not agents defined (SRV-1674)
- Meeting invitations are not sent to guests that are a user contact but don't belong to the user's domain (QUOBDEV-654)
- Meeting invitations are not sent if the user domain is written completely with uppercase letters (QUOBDEV-651)
- Calls transferred to a SIP destination caused a wrong entry in the conference log in parallel-SIP scenarios which also caused a crash in the iOS SDK (QSS-281)
- Calls were not correctly setup after a redirection from one audiomixer to another with parallel-sip configuration and HA setup (QSS-288, QSS-234)
- Users were allowed to set another username in uppercase as their alias (SRV-1610)
- Incoming SIP that matched a `wac-ims-credentials` that starts with a (+) symbol were rejected (QSS-179)
- Filters `createdAtBefore` and `createdAtSince` were not working as expected, as they used `LessThanOrEqual` or `GreaterThanOrEqual` filters rather than``LessThan`` and `GreaterThan` filters (SRV-1664)
- The call does not progress when calling to a ringing group and the first user of the group is offline (QSS-266)
- An ongoing call ends if the user logs out from another session (QSS-270)
- Making a call to a *user group* with a space in its ID gets no answer and ends after a timeout (QUOBDEV-694)

### 3.40.3 Known issues

- Service sippo-exporter crashes sending a GET to `api/queues` (DEVOPS-301)
- Presence remains as "busy" if a user is in a call and the connection is closed abruptly (ASLO-33)

### 3.40.4 Technical improvements

- Mobile push notification implementation has been completely redesigned in order to comply with the last iOS policies and to accomodate both hybrid and native applications. All push notifications are now sent from the sippo-server services, including chat notifications which were previously sent by the XMPP server.
- Unify user group events into a single one (group-updated)

### 3.40.5 REST API

#### 3.40.5.1 New features and changes

- REST API documentation is now described using the OpenAPI format.
- Added /kmanage/k8s/getlogs endpoint to gain get logs of an specific pod (`kubectl logs <pod-name>`)
- Added endpoint /kmanage/loglevel to set the log level of /kmanage endpoints between levels "Info", "Warn","Error","Debug" and "Trace" without service restart
- Added two endpoints to manage system backups:
  - `/kmanage/backups/getbackups`: lists all backup file available in the system
  - `/kmanage/backups/downloadbackups`: download available backup files
- Added endpoint /kmanage/reporting/getreport that gets reports of the last 24 hours.
- Added endpoint /kmanage/siponhold to obtain conference object from `roomId` and sent it to the audiomixer service to implement a SIP reINVITE
- Added several endpoint for the "Agent assignment" functionality:
  - `/kmanage/agentassigner/list` that gets every agent assigner entries
  - `/kmanage/agentassigner/delete/{id}` that deletes and agent assigner element
  - `/kmanage/agentassigner/showroute/{id}` that shows the route of this entry at the agent assigner
  - `/kmanage/agentassigner/addroute` that creates a new route to the agent assigner element
- Added endpoint UserContacts for better contact management.
- Added endpoint AddressBooks for better contact management.
- Added endpoint `/kmanage/template` to get swagger template file in JSON format
- Added several endpoints to gain visibility over the Kubernetes cluster status:
  - `/kmanage/k8s/clusterstate`: lists the cluster state (`kubectl get pods -o wide`)
  - `/kmanage/k8s/clusterservices`: lists the running services (`kubectl get svc --all-namespaces -o wide`)
  - `/kmanage/k8s/getlogs`: gets logs of a specific pod (`kubectl logs <pod-name>`)
- Added boolean field `isPasswordProtected` to the meeting object that indicates if a meeting is protected by a password.
- Added query parameter `userId` to filter meetings to which an specific user has been invited.
- Added endpoint `/kmanage/xmpp-server/getuserschat` that returns a list of users with sent chats and the number of sent chat by each user (1-by-1 chats and group chats)
- Added an endpoint to delete sent chats by an specific user (`/kmanage/xmpp-server/deleteuserchats`)
- Endpoint /xmpp/roster/{username}: replaced by full subset:
  - `xmpp/push`
  - `xmpp/push/register`
  - `/xmpp/push/unregister`
- Endpoint `/pushNotifications/{userId}` replaced by `` `/pushNotifications/byUserId/{userId}`:
- Added endpoint `/kmanage/getrecord` to list all recording files available in the system.
- Added endpoint `/kmanage/downloadrecord` to download a recording file.

- Added endpoint `/kmanage/usergroup/logout` to force logout of every participant of a given user group.
- Added endpoint `/kmanage/user/location` to obtain the location of a user.

#### 3.40.5.2 Bug fixes

- Information exposure through error message on `/sapi/o/token` (SEC-59)
- Unauthenticated user could query presence information via `/sapi/presence/wac-user` (SEC-61)
- Log message returned in response to endpoints credentials and domains (SEC-68)
- Potential denial of service on `/sapi/sessions` requests that results in "504 Gateway time-out" after around three minutes (SEC-62)
- Information leakage in `/sapi/filesharing` endpoint (SEC-69)

## 3.41 v4.1.0

### 3.41.1 New features

- Global improvements regarding the internal work of the system. The unified system based on a message-broker allows to add more services and remove them easily, giving more flexibility to the architecture. The main change comes from latency optimization, based on a ws-proxy that allows a unique system connection with better and improved performance.
- Regarding monitoring software it was enhanced with log view and troubleshooting filters navigation. No more consoles to extract logs. Just use the logs, cross metrical performance KPIs data with direct container logs and get your info easy and direct.
- Attended transfer (available only in the Javascript SDK in this version).
- TLS 1.3 support is included. Your apps are now more secure, more efficient, more reliable.
- NGINX was selected as ingress-controlled for the cluster, so some enhances here from the original Traefik media
- SIP early media support and more options regarding the codec selection and media bandwidth allocation for calls. Also were applied some audio improvements to reduce audio-noise on calls.
- Meetings now include more options to add flexibility to the meeting invites. REST API was improved to support more use cases
- Improved and resolved multiple issues with chat notifications.
- Ability to reach SIP endpoints for anonymous users.
- Add error logs in QSS when "resolvePush" is set to true and it is not possible to reach Push service in sippo-server
- CDR (call detail record) display and download in Sippo manager
- Integrate services logs viewer in Sippo manager
- Generic filter in /conferences/ endpoint to retrieve list of conferences
- Screen sharing support in the desktop application
- Allow more than 9 video participants on a conference
- Prevent certain types of files (JS, HTML and any potentially harmful type) from being transferred (filetransfer)
- Avoid credentials being leaked by SAPI endpoints

- CORS configuration mechanism
- Backend server resource reach-ability service
- Transition from Socket.io to plain Websocket
- Apply traffic security and anti-DoS measures to k8s ingress (nginx)
- Support to multiple audiomixers (high availability)
- Include the busy tone when call is rejected
- Background noise reduction techniques in SippoSDK-JS and webphone
- Provide audio toggle for mobile applications
- Ability to change the default bandwidth of the video flows
- Early media support with sippoMS.sfu
- Automatic SMS / email notification to users when creating a meeting
- Improve mail templates for more customization on body and headers
- Chat / call / SIP agent assigner (SDK, REST endpoint)
- Migrate XMPP storage to SQL based system
- Ability to enable/disable "conference chats" in Application Server
- Group chat push notifications -
- Restrict registration to a single endpoint
- Add REFER support on the SIP integration
- SIP/WebRTC: On hold music for held participants
- Add SIP header manipulation for AudioMixers

## 3.42 v4.0.0

The list below shows the overview of the main features added since the previous version at each category. A detailed list is provided in the next subchapter.

### 3.42.1 New features

- The internal architecture has been separated into Sippo AS and Sippo MS. Besides that, new features have been implemented as a number of microservices -interconnected by a high performance message broker- to improve resilience and scalability.
- On the client side, two new mobile SDKS have been added for iOS and Android operating systems.
- Sippo manager has been released to provide better logging and monitoring capabilities. Logs can also now be sent to an external ElasticSearch server for analytics and troubleshooting.
- Added delegated authentication against Microsoft ADFS servers and upgraded Google oAuth to work with newer Google versions
- Added a number of PBX-like features such as ringing groups, attended transfer, unattended transfer, onHold, etc... That means that Sippo wac can provide signalling capabilities on its own, without requiring a third-party PBX o SIP backend.
- A new service named "conferencelog" allows also to retrieve the conferences where a user has participated, including those where multiple participants are joining and leaving during the conference and also including low-level information on invited participants, rejected and accepted invites, transferred users, etc..

- Integration with external SIP backend has been also improved with a "SIP parallel leg" approach. That means that service providers can easily add collaboration to exiting voice-only networks.
- Room management is more powerful in this version, as it's possible to mix and match PSTN users, Sippo users and guest users in the same room. We've also changed the associated logic so room ownership is now handled in a more efficient way.
- Phonebook service has been added to better handle shared contacts and group contacts. In addition to personal contacts and domain contacts, administrators can now define a set of contacts that are assigned to specific users or groups. Users can also manage their own private phonebooks in addition to the ones created by the administrator. Improved mobile and directory synchronization of contacts is also included in this version.
- Deployment over Docker infrastructure is now fully supported and it's the recommended deployment choice for trials and PoCs. Automation has been done with Ansible scripts that don't require any manual installation. On the other hand, deployment over Kubernetes infrastructure is now the default choice for production environments. Deployments over public cloud are now also supported in Amazon AWS and Microsoft Azure.
- On the testing side, a number of load testing scripts and functionality testing scripts are available using third-party QA tools, in order to test the deployment before production.
- Sippo MS has been included into the architecture so a third-party WebRTC media server is now an optional choice. This media server can handle H264, VP8 codecs and is also able to interact with traditional voice networks using G711 codecs.
- This version also enables screen sharing from Chrome and Firefox without requiring a plugin (desktop only).
- Meetings have been refactored to make them more flexible and manageable, so that they can have a DDI assigned to access the meeting from PSTN or PBX (dial-in).
- This version incorporates a completely redesigned messaging service, that allows to have permanent chat between users, personal chat rooms, predefined chat rooms and other improvements. It also adds new user capabilities so administrators can allow or disallow messaging functions to users and user groups. That means for example that some user can have chat functions or are able to create groups, while others cannot do that.
- Mobile notification have been improved both on iOS Callkit and Android Callservice, with deeper integration with the existing javascript SDK and the new native SDKs. One to one Incoming chat messages are also notified with a push message.
- This version allows to configure codec specifications for stored recorded file. In addition, new integration options with storage services allows to actively notify about a file ready to be stored (or share with participants).
- Users can now be classified into groups (usergroups). That allows new use case such as to define ringing groups to call all participants of a predefined group, and also to have predefined phonebooks.
- Connector to AD/LDAP for oAuth authentication
- Ability to delegate authentication with Google oAuth
- Ability to delegate authentication with MS ADFS
- Domains can have default contacts that are searchable to every domain user
- Added abstraction layer to support different database schemas
- Ability to distribute Sippo AS and MS microservices into several hosts
- Ability to customize recording video container
- Recording support for h264 and vp8
- Added "SIP parallel leg" functionality to interop with SIP networks
- Sippo manager: user management (create, edit, update, delete)

- Sippo manager: dashboard with business KPI and system status graphics
- Send audio and video clips
- Ability to send DTMFs on the SIP channel on conferences (RFC2833 and RFC4733)
- Added "Agent assigner" service to interact with external services in contact-center scenarios.
- Toggle camera support for SippoSDK-JS and SippoSDK-iOS
- Ephemeral chat group for anonymous rooms
- Support for getNativeDisplayMedia: Firefox, Chrome
- Recording support for H264 and VP8
- REST conference status exposition
- Support to transfer of PSTN calls to other Sippo users or PSTN
- Improved security login mechanism
- Improved signalling headers to identify "to" field in parallel-SIP environments
- oAuth2 tokens can have now a expiration time and argument name
- Support to remote logging into an Elastic Search instance
- User alias enhances: editable by waccli and allow dots in users' aliases regular expression
- This version of the REST API has a number of new methods in order to allow external systems to interact with the new backend functionalites available in this realease, specially around conference management, user management and phonebook management.

CHAPTER

FOUR

# INSTALLATION

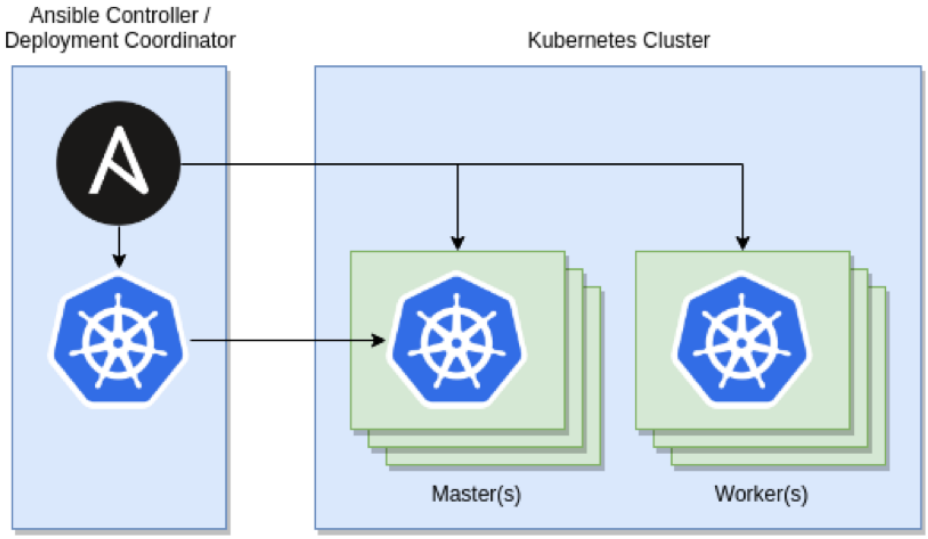
This section explains how to deploy the Quobis WAC from scratch or by upgrading from a previous version. Latest GA version as of 17 Apr 2024 at 10:00 is *5.0*. Release notes are available *here*.

## 4.1 New installation (v5.0)

This section explains the pre-requisites and installation steps to deploy **Quobis communication platform v5.0 from scratch**. The Quobis wac is software-only product, so there are not physical items to ship, everything can be downloaded from our servers.

The Quobis Communication Platform is a cloud native, containerized application that can be deployed on on-premises (bare metal or virtualized infrastructure), Private and Public Cloud environments. It is deployed on top of Kubernetes , also known as K8s, an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.

The actual installation and deployment is doing by leveraging Ansible , an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes. Ansible works by connecting to the destination machines nodes and pushing out small programs, called modules, to them.




The following chapters explain all the required steps to set up the platform.

### 4.1.1 Planning and pre-requisites


- *Choosing the deployment environment type*
- *Kubernetes cluster sizing*
- *Hardware requirements*
  - *On-premises deployments*
  - *Cloud deployments*
  - *Sizing of media server nodes*
  - *Sizing of application server nodes*
- *Network requirements*
  - *Firewall rules and security considerations*
  - *Sizing ports for the SIP media connections*
  - *Sizing ports for the WebRTC media connections*
  - *Sizing of media servers required according to traffic load*
- *Bandwith requirements*
- *Certificates*
- *Additional requirements*


#### 4.1.1.1 Choosing the deployment environment type

Quobis communication platform supports on-premise or cloud scenarios, from local and private deployment to global and multi location deployments depending on your organization needs.

- **On premises:** deployments use your own network, hardware and hypervisor. You will need to consider that compromised data will be stored in your private infrastructure with restricted access, providing a higher level of security. On premise deployments will need an appropriate server where to host the VMs and the proper network configuration. If you already have a mature IT network with servers in the appropriate locations, this may be a suitable option. If you want to install on-premises, you will need to install a hypervisor on your servers to manage the deployment and configure the VMs to setup the Kubernetes cluster. Quobis communication platform currently supports VMware hypervisor only.
- **Cloud deployments:** the supported public cloud provider as of today is Amazon Web Services (AWS). This option does not require hardware infrastructures, you just have to reach an agreement with the cloud service provider, supplying an easy and fast way to set up the environment to deploy the platform.

#### 4.1.1.2 Kubernetes cluster sizing

A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node which host the Pods, that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

The performance of the Quobis communication platform is determined by the number of nodes and the amount of resources of each node, in terms of CPU and memory. Depending on the number of nodes and the resources used by each of them the number of calls managed at time by the system may vary significantly. Additionally, Kubernetes provides to the cluster the capability to support High Availability,

which means a seamless service when a node breaks down. This scenario needs, at least, 3 master nodes and 2 worker nodes in order to balance the workload between the active nodes.

You also must take into account the deployment location to improve the conference's performance. The server should be located as close as possible to the main base or your organization due to the time issues. For instance, choose the AWS EU-region if your organization base is located in Europe.

Depending on the type of deployment, the number of current calls and the number of provisioned users, the number of nodes and their resources must be chosen accordingly.

- For a **high-availability** deployment, 3 master nodes and at least 2 worker nodes are required
- For a **standalone** deployment, 1 master node and at least 2 worker nodes are required

#### 4.1.1.3 Hardware requirements

There are no specific hardware or vendor constrains as Quobis Communication Platform can run on top of virtual machines (VMs). Just make sure that the hardware sizing is appropriate for the VW specified below.

##### On-premises deployments

- The selected hypervisor must be VWMare
- Supported OS: Debian 9 or higher
- Master nodes: at least 4 vCPU, 8 GB RAM, at least 100GB HDD
- Worker nodes: at least 8 vCPU, 16 GB RAM, at least 100GB HDD
- Two option for storage:
  - Highly available persistent storage for Kubernetes through Longhorn
  - A Network File System (NFS) server provided within the product to store all the persistent data used by the cluster.

##### Cloud deployments

- The selected hypervisor must be the AWS virtualization platform and EC2 instances
- Supported OS: Debian 9 or higher, Ubuntu 16 or higher
- Master nodes: AWS t3a.medium instances, 2 vCPU, 4 GB RAM, EBS storage, up to 5Gbps bandwidth
- Worker nodes: AWS c5.2xlarge instances, 8 vCPU, 16 GB RAM, EBS storage, up to 10Gbps bandwidth, up to 4750Mbps EBS throughput
- Storage: Amazon Elastic File System (EFS) standard is the solution used to provide persistent storage in Amazon Web Services deployments.

### Sizing of media server nodes

The Quobis platform deals with video and audio in different ways, as explained in detail in the *media management* section:

- The audio is managed in a *MCU* (Media Control Unit), also referred as *audiomixer*. The audio coming from all the participants is mixed here.
- The video is managed in an *SFU* (Selective Forwarding Unit). The video coming from all the participants is relayed to the rest of participants.

There are two services which consume CPU cycles in the media server nodes:

- Audiomixer: it consumes CPU mainly in two tasks:
  - Opus transcoding: the media coming from the WebRTC endpoints (browsers and mobile SDKs) needs to be transcoded before being mixed. Opus transcoding is a CPU-intensive task and the one which porcentually consumes more CPU cycles. If PCMU/PCMA (G711) is used instead of Opus, the CPU consumption is much lower.
  - Audiomixing: the mixing of audio, once transcoded, also requires CPU real-time processing.
- SFU: it consumes CPU to keep the media connections open between browsers/mobile apps and the SFU. These connections are called PeerConnections in WebRTC terminology. The number of open PeerConnections increases linearly with the number of simultaneous media sessions and pseudo-exponentially with the number of participants in each session.

Taking into account the factors above and an average distribution of users in conferences with different numbers of participants, we can consider the following sizing rules:

- One node (8vCPU) for each 200cc using Opus
- One node (8vCPU) for each 400cc using PCMU/PCMA.

### Sizing of application server nodes

Besides the media processing, the CPU consumption of the application pods must also be considered. The CPU processing will depend on several factors:

- the number of simultaneous active sessions
- the number of concurrent calls.
- the number of calls per secon
- use of the presence service. Presence can be an issue in large setups where each user receives an update with any presence change. So a huge average number of contacts per user will lead to a higher CPU utilization.
- provisioning processes through the REST API may lead to peak CPU usage.
- Sessions from browser-based apps that remain active while the web app is open.
- Sessions from mobile apps that keep the session active only while the app is in foreground.

Taking into account the factors above, we can consider the following sizing rules as a rule of thumb:

- one 8vCPU node per 1,000 simultaneous sessions when presence is enabled.
- one 8vCPU node per 2,000 simultaneous sessions when presence is not enabled.

Please consider the rules above as references, since the load may differ depending on the use cases. An active monitoring of the platform may is quite userful to adjust the number of nodes.

#### 4.1.1.4 Network requirements

##### Firewall rules and security considerations

Quobis Communication Platform needs communication between cluster nodes (master and workers) and also needs to expose different ports to allow voice and video calls. The following table summarizes the required network ports to configure:

Table 1: Networking requirements

| Req # | Origin | Destination | Destination IP | Destination ports | Protocol | Service | Encryption | Description |
|---|---|---|---|---|---|---|---|---|
| 1 | WebRTC clients | AS | AS IP | 80,443 | TCP | WSS | SSL | Application traffic |
| 2 | WebRTC clients | MS | IP GW-SFU | 20000-29999 | UDP | DTLS-SRTP | SSL | WebRTC media |
| 3 | WebRTC clients | MS | IP GW-TURN | 3478 | UDP | STUN | Unencrypted | TURN (NAT discovery) |
| 4 | WebRTC clients | MS | IP GW-TURN | 443 | TCP | CO-TURN | TLS | TURN (media relay) |

There are other rules that need to be applied only when there is a SIP trunk connection with an external SIP entity such as a PBX, PSTN via SBC, etc. (rules 5,6,7) or when the deployment includes mobile clients (rules 8, 9 and 10) as per the following table:

Table 2: Advanced networking requirements

| Req # | Origin | Destination | Destination IP | Destination ports | Protocol | Service | Encryption | Description |
|---|---|---|---|---|---|---|---|---|
| 5 | SIP trunk | MS | IP GW-audio | 5060 | UDP | SIP | Unencrypted | SIP signalling |
| 6 | SIP trunk | MS | IP GW-audio | 5061 | TCP | SIP | TLS | SIP signaling |
| 7 | SIP trunk | MS | IP GW-audio | 20000-29999 | UPD | RTP/RTCP | Unencrypted | SIP audio (optionally encrypted using SRTP) |
| 8 | Quobis AS | AS | AS | 5222 | TCP | XMPP | TLS | Messaging from mobile apps |
| 9 | WebRTC client | Apple APNS | 17.0.0.0/8 | 443 | TCP | HTTP/TLS | SSL | Mobile push notifications |
| 10 | Quobis AS | Apple APNS | 17.0.0.0/8 | 2195,2196 | TCP | HTTP/TLS | SSL | Mobile push notifications |

Note: rule #5 refers to the port used to exchange traffic between the SIP proxy and the core SIP network elements.

Note: rule #8 is needed because the messaging in mobile applications does not use Websocket but TLS over TCP connections to the server.

**Note:** Websocket connection timeouts: we minimize the amount of keepalive messages in order to optimize the connection setup time and the amount of traffic exchanged between our SDKs and the backend. To avoid connectivity problems, we recommend to setup traffic timeouts with high values

(3600s) in the reverse proxies (e.g. F5 solutions) between the clients and the Quobis WAC cluster. This way we prevent websocket connections (RFC6455) from being abruptly closed.

### Sizing ports for the SIP media connections

AS explained in the rule #7 above, a range of UDP ports need to be opened in order to allow media traffic from the SIP trunk connection. The following considerations must be taken into account:

- Non-ParallelSIP scenarios: the number of ports can be obtained as AverageNumConcurrentConferences * 2 (one port for RTP, one port for RTCP)
- ParallelSIP scenarios: the number of ports can be obtained as AverageNumConcurrentConferences * 4

### Sizing ports for the WebRTC media connections

As explained in the rule #2 in the table above, a range of UDP ports need to be opened in order to allow WebRTC media traffic from/to the WebRTC endpoints. As explained in the *media management* section, Quobis WAC uses a SFU (Selective Forwarding Technology) to deliver the video flows to all the participants in a video conference. That means that the media servers will send an individual flow of the video published by each participant to each and one of the participants of the conference, simultaneously. Taking this fact into accoun, the number of used ports will depend on two variables:

- the number of simultaneous users of the platform
- the average number of participants at each conference

The formula used to calculate the ports used in a single conference can be computed from the following paramenters:

- NumParticipants = Number of participants in the conference.
- NumPortsPerConference = Total number of used UDP ports used in a conference.
- NumPortsVideoSubscribers = Number of UDP ports used for video per conference to receive video from the SFU.
- NumPortsVideoPublishers = Number of UDP ports used for video per conference to send video to the SFU.
- NumPortsAudio = Number of UDP ports used for audio per conference. For audio the same port is used to publish and subscribe.

and applying this formula:

- NumPortsVideoSubscribers = (NumParticipants - 1) * NumParticipants
- NumPortsVideoPublishers = NumParticipants
- NumPortsAudio = NumParticipants
- NumPortsPerConference = NumPortsVideoSubscribers + NumPortsVideoPublishers + NumPortsAudio

Please note that this formula only applies if all the participants are publishing video and the rest of participants are subscribed to it. This represents the worst case in terms of port usage. If a user is neither subscribed nor publishing video those ports should not be considered in the formula. Please note that this refers to the number of open ports per media server, if your architecture includes several media servers then the number of ports must be distributed amongst them. As a reference, in the table below you can see the correspondence between number of participants in a conference and the number of ports used:

Table 3: Number of ports used in a conference room

| Participants in the conference | Ports used in the conference |
|---|---|
| 2 | 6 |
| 3 | 12 |
| 4 | 20 |
| 8 | 72 |
| 16 | 272 |
| 32 | 1056 |
| 64 | 4160 |

As it can clearly seen, the figures are exponential and for conferences above 32 participants, with all of them publishing video at the same time, the number of ports is huge and so will be the used bandwidth. We can perform an estimation to calculate the number of open ports that we need which takes into account the number of **simultaneous** users using the service and the **average** number of participants at a conference. This will give a good estimation of the number of ports which need to be configured to avoid any service disruption if the limit is reached. So to calculate the right number of UDP ports we need to open please consider the following definitions:

- SimultaneousUsers = number of users connected to a conference at the same time (do not confuse with the number of registered users).
- AverageNumConferenceParticipants = average number of participants in a conference.
- AverageNumConferences = average number of conferences.
- TotalPorts = Total number of ports required by the system.
- NumPortsPerConference = number of ports per conference calculated using the formula previously defined.

and compute the following formula:

- AverageNumConferences = ceil (SimultaneousUsers / AverageNumConferenceParticipants)
- TotalPorts = NumPortsPerConference(AverageNumConferenceParticipants) * AverageNumConferences

As a reference, the table below shows the number of estimated ports required depending on AverageNumConferences and SimultaneousUsers (calculations per media server):

Table 4: Number of ports used in a conference room

| SimultaneousUsers | Average Num Conference Participants | Total ports used |
|---|---|---|
| 100 | 4 | 500 |
| 100 | 8 | 900 |
| 500 | 4 | 2500 |
| 500 | 8 | 4500 |
| 1000 | 4 | 5000 |
| 1000 | 8 | 9000 |

The figures above should give a conservative value. However if a shortage of ports is detected in high load conditions the range must be increased. If the use of such a wide range of ports can mean a problem then you should consider the use of a TURN server which can multiplex the media coming and going from/to different endpoints on a single UDP port. However using UDP would be the preferred option in terms of simplicity and performance.

#### Sizing of media servers required according to traffic load

In summary, according to the selected MS worker nodes referred above, each of them must open the indicated number of ports depending on the audio codec in order to cover the main available scenarios:

- 2000 opened port per SFU + MCU (Opus) -> 200 cc per Media Server
- 4000 opened port per SFU + MCU (G711)-> 400 cc per Media Server

#### 4.1.1.5 Bandwith requirements

Administrators need to take into account how much bandwith is going to be required to cope with the application traffic, specially for video applicatons. This calculation is directly related with the codecs in use and the quality allowed for each video stream. Please check the *media quality* section to understand the codec choice and how much badwith will be required in order to provide a good quality of service.

#### 4.1.1.6 Certificates

Digital certificates are needed in the system in order to complete the installation. You need to provide:

- TLS certificates for the ingress controller (mandatory). More information can be found in the Nginx documentation site
- Apple push notification services (only when mobile devices are included in the deployment).

#### 4.1.1.7 Additional requirements

There are other additional requirements that need to be fulfilled before starting the installation process:

- Remote access must be guaranteed to master and worker nodes during the installation process.
- Internet access is needed in all the nodes during the installation procedure in order to download external dependencies and container images. Once the Quobis Communication Platform is installed and running, the internet connection can be blocked if needed.
- Quobis Registry credentials (username and token) is needed to download the docker images during installation and update procedures from registry.quobis.com and registry.gitlab.com
- Gitlab credentials or accees token to download the repository with the installation sofware located at https://gitlab.com/quobis/devops/sippo-k8s
- Domain and valid certificates: a public domain and the SSL certificates have to be available to access the Quobis Communication Platform from a browser.
- Public IP addresses: each Media Server (MS) worker needs a public IP address in order to manage video and audio of WebRTC calls. The IP traffic can be forwarded from an external firewall.

### 4.1.2 Installing Ansible

**Contents**


- *Basic Ansible concepts*
  - *Control node*
  - *Managed nodes*
  - *Inventory*
  - *Tasks*

  - *Playbooks*
- *Requisites*
- *Installing Ansible on Ubuntu*
- *Installing Ansible on Debian*
- *More information*


Ansible is an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes. Ansible works by connecting to the destination machines nodes and pushing out small programs, called modules, to them.

#### 4.1.2.1 Basic Ansible concepts

These concepts are common to all uses of Ansible. You need to understand them to use Ansible for any kind of automation.

##### Control node

Any machine with Ansible installed. You can run Ansible commands and playbooks by invoking the ansible or ansible-playbook command from any control node.

##### Managed nodes

The network devices (and/or servers) you manage with Ansible. Managed nodes are also sometimes called "hosts". Ansible is not installed on managed nodes.

##### Inventory

A list of managed nodes. An inventory file is also sometimes called a "hostfile". Your inventory can specify information like IP address for each managed node.

##### Tasks

The units of action in Ansible. You can execute a single task once with an ad hoc command.

##### Playbooks

Playbooks are ordered lists of tasks, saved so you can run those tasks in that order repeatedly. Playbooks can include variables as well as tasks. Playbooks are written in YAML and are easy to read, write, share and understand. Playbooks tell Ansible what step the user wants to execute on a particular machine in a repeatable and re-usable way, well suited to deploying complex applications. They are like a to-do list for Ansible that contains a list of tasks.

#### 4.1.2.2 Requisites

Quobis Communication platform installation requires a machine from where all Ansible and other script commands are issued ((which could easily be a laptop). This machine where Ansible is installed is called the "control node" and can be part of the cluster or a separate dedicated machine. The idea behind this is to have a unique element where the support access is centralized. No direct access to the working cluster is required, just access to this working station.

Check the following requirements on the machine where you will to run the Ansible installer:

- Supported OS: Debian 9 or higher and Ubuntu 16.04 or higher
- At least 2GB of RAM
- Ensure that all the target machines have network visibility with the Ansible script installer and are reachable via a SSH connection.
- Includes sudo and an accessible user included on sudoers: `sudo usermod -aG sudo <host_username>`

#### 4.1.2.3 Installing Ansible on Ubuntu

To configure the PPA on your machine and install Ansible run these commands:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo apt-add-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Note: On older Ubuntu distributions, "software-properties-common" is called "python-software-properties". You may want to use apt-get instead of apt in older versions. Also, be aware that only newer distributions (in other words, 18.04, 18.10, and so on) have a -u or –update flag, so adjust your script accordingly.

#### 4.1.2.4 Installing Ansible on Debian

Debian users may leverage the same source as the Ubuntu PPA.

Add the following line to /etc/apt/sources.list:

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main
```

Then run these commands:

```
 $ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
 $ sudo apt update
 $ sudo apt install ansible
```

Note: This method has been verified with the Trusty sources in Debian Jessie and Stretch but may not be supported in earlier versions. You may want to use apt-get instead of apt in older versions. Further documentation about Ansible installation can be found in the Ansible website.

#### 4.1.2.5 More information

In case of any issue with the Ansible instalation, you can check the official installation guide that can be found here: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

### 4.1.3 Ansible configuration

Once that the Ansible package is installed in the control node (the machine that will be used for the installation), the next step is to download the Quobis wac installer.

#### 4.1.3.1 Getting the Quobis wac installer

The Quobis wac installer can be obtained in two different ways:

- If you have access to a Quobis Gitlab account, you can clone the repository with the following command that will create a directory name "k8s-installer" with the installer files:

```
$ git clone --depth 1 --branch 2.35.0  https://gitlab.com/quobis/devops/k8s-
↪installer.git  # use this branch for v5.0.0
```

- Otherwise, please contact your support team and you'll get a ZIP file containing the same source code and version.

Once you have the files in you computer, this is the content that you'll see inside the "k8s-installer" directory. The "ansible" folder contains the host.conf file, the playbooks and the vault file. The "kubernetes" folder contains the Kubernetes manifests that are used during the installation.

```
$ ls -l
-rw-r--r--   1 owner  staff  156583 23 dic 17:40 CHANGELOG.md
-rw-r--r--   1 owner  staff     483 23 dic 17:40 README.md
drwxr-xr-x  14 owner  staff     448 23 dic 17:40 ansible
drwxr-xr-x  61 owner  staff    1952 23 dic 17:40 kubernetes
```

#### 4.1.3.2 Setting up the inventory

Now it's time to setup the inventory that tells Ansible where the managed nodes are located. First, you need to copy the "default" inventory to take it as reference:

```
$ cp -r k8s-installer/ansible/inventory/default k8s-installer/ansible/inventory/<your_
↪company_name>
```

Now, remove the default passwords from the vault inventory in order to add the new ones (replace <your_company_name> with your inventory name):

```
$ cat /dev/null > k8s-installer/ansible/inventory/<your_company_name>/group_vars/all/
↪vault.yml
```

After that, you need to set up the cluster hosts's IPs and credential to allow Ansible to execute the automated tasks. You will find a YAML file with all entries enclosed in brackets [] that are referred to as "hosts". On the Kubernetes deployment you will find these five types of hosts:

- [master] The machines where the `kubectl` will run. All the cluster management is done from these machines, as they are the control panel of the deployment. Three instances are required when deploying on VMware.
- [node] Working nodes, the ones that do the job. They handle traffic, run services, etc. The minimum number of working nodes is three.

- [nginx] If your system requires an external entry point, this is it (only for on-prem, AWS uses the ELB service)
- [nfs] Only on onPremise deployments, it will handle a distributed storage system (AWS uses the EFS service)
- [turn] Server where the TURN services will be deployed.

For each host you must indicate:

- The node's IP address for SSH connection (reachable from Ansible control node)
- `ansible_user user`: name used by Ansible to log into the remote machine.
- `ansible_ssh_pass`: user pass used by Ansible to log into the remote machine.
- `ansible_sudo_pass`: sudo pass needed by the `ansible_user` to log as administrator on the remote machine

**Note:** Use single quotes to enclose values that include spaces, such as passwords or file locations

The hosts file can be edited with any available text editor, such as "vi":

```
$ vi k8s-installer/ansible/inventory/<your_company_name>/hosts
```

This is the default content of the k8s-installer/ansible/inventory/<your_company_name>/hosts file. You need to fill in the appropriate IP address for each virtual machine:

```
[master]
put_here_master_IP ansible_user=<user> ansible_ssh_pass=<password> ansible_sudo_pass=
↪<password> ansible_ssh_private_key_file=/home/user/.ssh/key.pem ssh_key_file='~/.ssh/id_
↪rsa'

[node]
put_here_node_1_IP ansible_user=<user> ansible_ssh_pass=<password> ansible_sudo_pass=
↪<password>
put_here_node_2_IP ansible_user=<user> ansible_ssh_pass=<password> ansible_sudo_pass=
↪<password>

[nginx]
put_here_nginx_IP ansible_user=<user> ansible_ssh_pass=<password> ansible_sudo_pass=
↪<password>

[nfs]
put_here_nginx_IP ansible_user=<user> ansible_ssh_pass=<password> ansible_sudo_pass=
↪<password>
```

#### 4.1.3.3 Installing the certificates

Now, you need to provide the digital certificates for the following services:

#### Certificates for the ingress controller

We created a folder on the Ansible installer that will help you to have all ready before start. Save the following files in the k8s-installer/ansible/ssl/ folder:

- `certName.crt`, `certName.key` files: public and private certificates to be used on the Nginx public interface, as this is the main entry point.
- `APIcertName.crt`, `APIcertName.key` files: public and private certificates to be used on the Nginx internal interface. Any self signed certificates will do the work. The sippo-server service uses it on the `wiface1` interface.

#### Certificates for push notification services

Save the following certificates in the kubernetes/prosody/configmaps/conf/certs.d/ path:

- apn.cer, apn.crt.pem, apn.key and apn.key.pem files: public valid certificates for Apple push services.
- voip_services.cer, voip_services.cer.pem files: public valid certificates for Apple VoIP services.

### 4.1.3.4 Vault management

Passwords are stored and encrypted in the `group_vars/all/vault.yml` vault file. Some of them must be introduced in this file manually. Additionally, the remaining passwords can be included in this file following this steps:

First, create a master password for your environment. We recommend storing that password in a password manager like LastPass or Bitwarden.

Next, create a new file named `vault-key` in `inventory/<inventory name>/`. This file must contain your master password.

Once done, you can create passwords for each service with the next command:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "passwords-deploy"
```

Once the passwords-deploy task has finished, and your vault file is populated with new random passwords, you can delete or leave the `vault-key` file.

Additionally, you need to include the dockerPassword with the registry credential provided by Quobis.

```
$ ansible-vault encrypt_string --vault-id <inventory name>@prompt '<registry-token>' --
↪name 'dockerPassword' >> <inventory name>/group_vars/all/vault.yml
```

### 4.1.3.5 Playbook configuration

Now it's time to configure the Ansible playbooks. Playbooks are the files where Ansible code is written.

The Quobis wac has all the available configuration files into `k8s-installer/ansible/inventory/<your_name_company>/group_vars/all` folder.

You need to change the value of some parameters according to your deployment needs. The following tables explain which ones of these values must be changed and how.

### Registry credentials

Table 5: Docker credentials

| Parameter | Value | Comment |
|---|---|---|
| registryServer | registry.gitlab.com | Use GitLab registry |
| registryNamespace | quobis | Do no change |
| dockerUsername | (blank) | Choosen username |
| dockerEmail | (blank) | Choosen email |

### Deployment options

Table 6: Configuration parameters

| Parameter | Value | Comment |
|---|---|---|
| infrastructure | 'omprem' | Type of infrastructure in deployment. Options: 'amazon' or 'onprem' |
| persistent | true | You can select if your infrastructure will have persistent data (databases, upload files, etc) |
| orchestrator | "k8s" | You can select the type of orchestrator to deploy de cluster. Options: k8s (production) and k3s (lab only). |
| maxStorageQuota | '60Gi' | Maximum disk space to keep databases, shared files, list of domains in messaging server and backups. |
| syncDate | true | True if containers' synchronization will be done from host machines |
| timeZone | /usr/share/zoneinfo/Europe/ | If syncDate is true, the timezone selected from host machines to be synchronized |

### Container versions

This is the list of the version of each container for version 5.0.0:

Please note that the format changes where we can have repeated values, such as in the case of the parameter "taggedMonitoringMachines". In this case, the value name start with a hyphen (-):

```
tagged: false

taggedSippoMachines:
    - name: kubernetes-node1
    - name: kubernetes-node2

taggedMonitoringMachines:
    - name: kubernetes-node1
```

### Nginx

This configuration is only required for on-premises deployments:

| Parameter | Value | Comment |
|---|---|---|
| loadbalancer | true \|\| false | |
| serverName | <server_name> | If true, Nginx-ingress will be deployed as Load-Balancer |
| nginx_key | <SSL_certificate_name>.key | |
| nginx_crt | <SSL_certificate_name>.crt | |
| nginx_sapi_key | <SSL_certificate_name>.key | |
| nginx_sapi_crt | <SSL_certificate_name>.crt | |
| ingressListening-Port | 32639 | Port where http_proxy is listening (internally). No need to expose it externally |

### WAC

| Parameter | Value | Comment |
|---|---|---|
| defaultSip-poServerAuth | true | If true, enable default WAC oAuth authentication |
| dev | false | If true, include static DNS server name entry in WAC container |
| mailHost | 192.158.1.39 | IP of the email server |
| mailPort | 25 | Por of the email server |
| mailFrom | change@me.com | Mail that Quobis WAC uses to send emails |
| meetingLinkHost | https://<server_name> | |
| meetingPath | m | Meetings URL path, defaults to https://servername/m/meeting-id |
| pstnExtension | | PSTN-reachable dial-in number to join a Quobis WAC meeting in E.164 format |
| pushActivated | false | If true, mobile push notifications in mobile apps are activated |
| startUpTimeout | 15 | WAC start up timeout in seconds. |

### Quobis collaborator

The following parameters are only required when installing Quobis collaborator. If you deployment the system and don't plan to deploy Quobis collaborator, you can leave them blank.

| Parameter | Value | Comment |
|---|---|---|
| customWebphoneStaticsSounds | false | Custom Sounds |
| customWebphoneStaticsI18n | false | Custom icons |
| customWebphoneStaticsFonts | false | Custom fonts |
| customWebphoneStaticsImages | false | Custom image |
| showContentInfo | true | Show instructions to the end user about how to use Quobis collaborator |
| webphoneDefaultSources | wac | Select the default contacts sources |
| webphoneDocumentTitle | <name of your system> | Select the HTML title of the document |
| webphoneDefaultDomain | quobis | Allows to enforce a domain in the login process. Cannot be left blank as it's used also for anonymous users |
| webphoneOauthRedirectUri | "{{ serverName }}/o/adfs/callback" | URL to redirect the oauth login to oauth server when using oauth2proxy |

### QSS

The following table lists the configuration parameters of *the signaling server (QSS)*:

| Parameter | Value | Comment |
|---|---|---|
| resolveGroups | false | |
| legacyUserGroupResolution | false | |
| onlyOneDeviceWithInvitations | false | |
| activateParallelSip | false | When true, parallel SIP routing is activated |
| voiceMailExtension | 999 | *VoiceMail* extension number |
| setOwner | false | |
| closeRoomWhenOwnerLeaves | false | |
| updateRoomOnwerWhenLeaves | false | |
| notInACall | false | |
| userIdAsRoomId | false | |
| allowInvitesWhileInCall | true | When true, incoming calls can be retrieved while on call |
| participantLimit | 10 | Set the participant limit on a same call |
| logChannelRepository | memory | |

### KeyCloak

| Parameter | Value | Comment |
|---|---|---|
| keycloak-DatabaseUser-name | keycloack | Set the database user |
| keycloak-DatabaseName | keycloak | Set the database name |

### Kitter

| Parameter | Value | Comment |
|---|---|---|
| realm | quobis | |
| welcomeEmail | true | When true, an email is sent to the created user. Set to false to avoid this behaviour. |
| updateUserDo-main | true | |
| phoneNumberRe-quired | true | |

### XMPP server

The XMPP server can either use internal storage or an external database (recommended). If `xmppDatabase` is set to true. PostgresSQL will be used and a valid version must also be entered.

| Parameter | Value | Comment |
|---|---|---|
| maxHistoryMes-sages | 50 | |
| mucLogExpire-sAfter | never | |
| xmppDatabaseV-ersion | 12.1-alpine | Set PostgreSQL database versión. Does not need to be changed |
| xmppDatabase | true | |
| xmpp-DatabaseUser-name | prosody_user | Username of the XMPP database. Does not need to be changed otherwise changes are made in the XMPP server |
| xmppDatabase-Name | prosody | Name of the XMPP database. Does not need to be changed otherwise changes are made in the XMPP server |
| xmppTLS | false | Enable XMPP messaging over TLS |
| xmppPort | 5222 | |

### Ingress

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Includes two network protection related features: Anti-DoS protection and CORS.

| Parameter | Value | Comment |
|---|---|---|
| exposeNginx-Ingress | false | |
| assigToNode | true | Deploy nginx-ingress in nodeNginxIngress |
| nodeNginxIngress | ops-ci-node2 | |
| commonName | '*.quobis.com' | Must go between quotes |
| wildCard | true | If true, commonName must have format of wildcard certificate |
| ddosProtection | false | When true, a DDOS protection is activated |
| rps | 5 | If DDOS protection is activated, maximum request per second allowed to all endpoints |
| rpsHigh | 20 | If DDOS protection, maximum request per second to Quobis collaborator endpoints |
| enableTeal | false | Teal deployment blue/green |
| cookieName | canary | |

### Certificates

These parameter are required only when using the "Let's encrypt" https://www.letsencrypt.org/. Let's Encrypt is a free, automated, and open certificate authority (CA), run for the public's benefit. It is a service provided by the Internet Security Research Group (ISRG).

| Parameter | Value | Comment |
|---|---|---|
| letsEncript | false | |
| awsRegionLetsEcript | <amazon_regin> | |
| provider | (blank) | |
| acmeDomains | (blank) | |
| letsEncriptEmail | (blank) | Email to be used in the LetsEncrypt account |

### Media server

| Parameter | Value | Comment |
|---|---|---|
| codec | "vp8" | |
| sfuWrapper: labelWrapper | sfu1 | Label for the wrapper, at least one is needed. Defaults to "sfu1" |
| sfuWrapper: ipJanus | 10.1.21.88 | IP of the Janus SFU |
| sfuWrapper: name | audiomixer | |
| sfuWrapper: node | ms1 | |
| asteriskVersion | 18.2.0 | Version of the Asterisk service used as audiomixer. |
| confbridgeTimeout | 28000 | |
| janusVersion | 0.9.5 | |
| mediaServersInK8s | falase | Deploy Asterisk, Janus and Kamailio in K8s cluster |
| webRtcIp | 20000 | Exposed Janus public IP to media stream |
| webRtcStartPort | 20499 | Exposed Janus start port to media stream |
| webRtcEndPort | 21000 | Exposed Janus end port to media stream |
| taggedMediaMachines.name | kubernetes-janus | |
| taggedMediaMachines.tag | ms1 | |

### SIP integration

| Parameter | Value | Comment |
|---|---|---|
| announceOnlyUser | false | Announcement if the user is alone in the conference room |
| rtpIp | 10.1.21.93 | Audiomixer external media IP |
| rtpStartPort | 10000 | Audiomixer start port for RTP port range |
| rtpEndPort | 10249 | Audiomixer start port for RTP port range |
| kamailioVersion | 5.4.4 | Kamailio and registry image version |
| sip-proxy.name | sip-proxy-1 | Kamailio deployment pod name |
| sip-proxy.node | ms1 | Kamailio deployment node |
| sip-proxy.sipIp | 10.1.21.93 | Kamailio deployment IP |
| sip-proxy.sipIpPrivate | 10.1.21.93 | Kamailio deployment private SIP IP |
| sipIp | 10.1.21.93 | Kamailio external SIP IP |
| sipPort | 5060 | Kamailio external SIP port |
| pbxIp | 10.1.3.2 | External endpoint SIP IP |
| pbxPort | 10.1.3.2 | Customer endpoint SIP port |
| pbxDomain | | R-URI, FROM and TO domain to requests |
| sipRegistration | false | Send REGISTER and authenticated REGISTER requests from Kamailio to PBX |
| sipAuthentication | false | Authenticate INVITE and REFER from external PBX to Kamailio |
| MWI | false | Message Waiting Indicator |
| multiLocation | false | |
| location.url: | (blank) | |
| location.group | (blank) | |

An example of a High-Availability SIP proxy configuration follows:

```yaml
sipProxy:
 - name: sip-proxy-1
   node: ms1
   sipIp: 10.1.21.93
   sipIpPrivate: 10.1.21.93
 - name: sip-proxy-2
   node: ms1
   sipIp: 10.1.21.96
   sipIpPrivate: 10.1.21.96
```

### TURN server

The TURN server is installaed as a service, not in the Kubernetes cluster.

| Parameter | Value | Comment |
|---|---|---|
| turnServerEnabled | True | When true, the TURN server is installed |
| turnDomain | (blank) | TURN server public domain |
| turnInternalIP | (blank) | Internal media server IP |
| turnPublicIP | (blank) | TURN server media server IP |
| turnPort | 443 | Port to be used by the TURN server |

### Recording

| Parameter | Value | Comment |
|---|---|---|
| enableRecording | False | |
| enableEncryption | False | File publickey.acs must be included in kubernetes/recording |
| encryptionMail | (blank) | |
| recordingVersion | 3.5.2 | |
| recordingType | none | Must be one of [none, video, audio, all] as explained in the *recording* section |

### Kubernetes API

| Parameter | Value | Comment |
|---|---|---|
| kapiVersion | 1.6.2 | |
| enableKapiui | True | |
| extended_onhold | false | |
| seconds_to_bye | 3 | |

### Teal service

Please note that the password for this configuration must be defined in the vault, not here.

| Parameter | Value | Comment |
|---|---|---|
| tealVersion | 1.2.0 | |
| tealDatabase-Name | teal_prod | |
| teal-DatabaseUser-name | teal | |

### Storage

The first two parameters applies for on-prem installation only, the following ones apply for the AWS configuration only.

| Parameter | Value | Comment |
|---|---|---|
| nfsServer.ip | 10.1.21.27 | |
| nfsPath | "/home/nfs" | |
| ef-sServer.efsFileSyste | fs-XXXXXXXXXXXXXXX | |
| ef-sServer.awsRegion | eu-west-3 | |
| efsProvisionerVer-sion | v2.4.0 | |
| efsPath | "/home/nfs" | |

### Database

| Parameter | Value | Comment |
|---|---|---|
| mongoAtlas | False | If true, the Mongo Atlas service is used instead of a local one https://www.mongodb.com/es/cloud/atlas |
| enableMongoOperator | true | If true, the database is deployed inside the cluster |
| replica | True | For multiple instances of the database deployment set replica:true and use a comma-separated array on the databaseUrl parameter |
| databaseVersion | 5.0.0 | Mongo database version (Does not apply to Mongo Atlas) |
| wacDatabaseName | wacDev | Set the WAC database name |
| qssDatabaseName | qss | Set the QSS database name |
| dispatcherDatabaseName | dispatcher | Set the Dispatcher database name |
| recordingDatabaseName | recording | Set the Recording database name |
| irsDatabaseName | irs | Set the IRS database name |
| mongoAtlasUser | <your_Mongo_Atlas_usernan | Set it when using a Mongo Altas database |
| mongoAtlasDomain | <your_Mongo_Atlas_domain | Set it when using a Mongo Altas database |
| mongoOpsUser | <your_Mongo_operator_user | Set it when using a Mongo operator |
| mongoOpsDomain | <your_Mongo_operator_dom | Set it when using a Mongo operator |
| noDnsResolution | false | Database hosted in external servers |
| databaseUrl | "quobis:{{ mongoPassword }}@database-mongo-0.database-mongo-svc.{{ namespace }}.svc.cluster.local | If the hostname of the machine do not resolve via DNS, it is needed to add it in each container, so set to true |
| databases.hostnam | (blank) | Hostname of the Mongo database (eg: mongo1.internal.quobis.com) |
| databases.ip | (blank) | Hostname of the Mongo database (eg: mongo1.internal.quobis.com) |

### Cluster maintainer

| Parameter | Value | Comment |
|---|---|---|
| maintainerVersion | 1.2.9-kubernetes1.15.3 | |
| deleteConferences | True | Delete conferences with state finished and older than CALL_HISTORY_DAYS days |
| callHistoryDays | 1 | Used to delete old conferences (see line above) |
| timeToPurgeChats | 6 | Purge prosody chats older than {{ timeToPurgeChats }} in months |
| timeToPurgeAnonymousUser | 8 | Purge anonymous users older than {{ timeToPurgeAnonymousUsers }} in hours |
| timeToKeepChannels | 8 | Delete channels older than X hours |
| timeToRemoveInvites | 8 | Delete invites older than X hours |
| timeToRemoveConferneceState | 8 | Delete conferenceStates older than X hours |
| deleteResolveSubscription | true | Delete resolveSubscription collection |
| deletePresenceSubscription | true | Delete presenceSubscription collection |
| deletePushSession | false | Delete push session |
| timeToKeepPushSession | 8 | Delete push session older the X days |
| scheduleCron | 0 * * * * | Must be enclosed between quotes |
| enableBackup | True | |
| enableDisasterRecovery | False | |
| backupFolder | (blank) | Name of backupFolder available to be used in the NFS or EFS storage |

### Quobis manager

| Parameter | Value | Comment |
|---|---|---|
| grafanaHost | (blank) | |
| grafanaUser | (blank) | |
| grafanaVersion | 8.2.7 | |
| lokiVersion | 1.4.0 | |
| prometheusVersion | v2.22.2 | |
| managerVersion | 1.6.1 | |
| managerUsername | qa@quobis | It's advisable to create a different user (admin role) for Quobis Manager to avoid sharing sippo-server admin credentials |
| sippoExporter | 1.6.0 | |
| support_sql_email | (blank) | |
| DBDROPDAYS | 7 | Homer database rotate period |
| hepServerIp | (blank) | HEP host IP |
| hepNode | (blank) | HEP host name |

### 4.1.4 Install the Kubernetes cluster

**Note:** This step is not required if you already have a Kubernetes cluster up and running. In such a case, skip to the next step: <install_deployment>.

Once all the required configuration variables are set in the playbook, it's time to let Ansible install the software into the target machines according with our configuration. This is achieved by using the ansible-playbook command, which runs the Ansible playbooks, executing the defined tasks on the targeted hosts according to a set of tags.

The first step is to install a Kubernetes cluster in the target machines. It involves the use of tags in the ansible-playbook. A tag is an attribute that you can set to an Ansible structure (plays, roles, tasks), and then when you run a playbook you can use –tags or –skip-tags to execute a subset of tasks. In other words, tags are a way of telling Ansible which tasks it should perform and which not.

The first step is to install the Kubernetes software and launch the cluster:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "deploy-cluster"
```

At this point, you should be able to connect to the machines where the cluster is deployed and test that the cluster has been correctly installed. In order to do so, you can enter the following command in the remote master node. If the cluster has been correctly installed, you should see the following result (please note that the node name might be different according to the name of the virtual machines where the deployment has been made)

```
$ kubectl get node
NAME                STATUS   ROLES                  AGE     VERSION
quobis-master       Ready    control-plane,master   1d      v1.21.14
quobis-node-1       Ready    <none>                 1d      v1.21.14
quobis-node-2       Ready    <none>                 1d      v1.21.14
quobis-node-3       Ready    <none>                 1d      v1.21.14
```

Once the cluster is up and running, you need to upload the kubernetes folder to the remote master node:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "upload-files"
```

Now it's time to install the Docker registry credentials in order to be able to download the software images:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "create-registry-secret"
```

In order to check that the token has been correctly installed, you can issue the following command and check that the result looks like this:

```
$ kubectl get secrets
NAME                                TYPE                                  DATA   AGE
registry-credential                 kubernetes.io/dockerconfigjson        1      1d
```

For on-prem deployments, you need to choose between NFS or Lonhorn:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "nfs-deploy"
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "nfs-client-provisioner-
↪deploy"
```

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "longhorn-deploy"
```

For AWS deployments, install the EFS services instead:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "efs-deploy"
```

In order to check that the EFS deployment has been correctly installed, you can issue a "kubectl get pods" command and check that you see the following pod:

```
$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS   AGE
(... other pods ...)
nfs-client-provisioner-7c7bd78d56-q998g     1/1     Running   0          116d
```

For on-prem deployments without a Load Balancer, you may want to install MetalLB:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "metallb-deploy"
```

Now that the K8S cluster is ready, you can move to the next step to deploy the Quobis wac software into the cluster.

### 4.1.5 Deployment

Now that the Kubernetes cluster is ready, it's time to let Ansible install the software into the target machines according with our configuration. This is also achieved by using the ansible-playbook command, which runs the Ansible playbooks, executing the defined tasks on the targeted hosts according to a set of tags.

A tag is an attribute that you can set to an Ansible structure (plays, roles, tasks), and then when you run a playbook you can use –tags or –skip-tags to execute a subset of tasks. In other words, tags are a way of telling Ansible which tasks it should perform and which not. In our case, the tag name explain what it does so it's quite straightforward to understand what you are doing.

Please note that you don't need to install all the tags listed below, as your deployment might not require all of them. For example, if you don't plan to deploy a SIP integration, you can skip it.

Create the PV and PVC needed to the cluster:

```
$ ansible-playbook -i inventory/<inventory name> main.yml --tags "persistent-deploy"
```

If the database is included in the Kubernetes cluster, you need to install the following tag as well:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "database-cluster-operator-deploy"
```

Deploy the core services required in any deployment:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "message-broker-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "kitter-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "erebus-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "storage-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "qss-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "sfu-dispatcher-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "sfu-wrapper-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
```

(continues on next page)

(continued from previous page)

```
↪yml --tags "postgresql-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "xmpp-server-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "keycloak-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "wac-core-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "sfu-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "audiomixer-deploy"
```

Deploy Quobis collaborator, if required:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "webphone-deploy"
```

Deploy Quobis manager, if required:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "manager-deploy"
```

Deploy Nginx-ingress to have access to the cluster:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "ingress-nginx-deploy"
```

Deploy the cluster maintainer:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "sippo-maintainer-deploy"
```

Just in those scenarios where SIP integration is needed, you must deploy the SIP proxy.

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "sip-proxy-deploy"
```

Now that the recording storage is ready, you can deploy the recording service:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "recording-deploy"
```

Kubernetes API deployment:

```
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "kapi-deploy"
```

Now you can deploy all the pods related with the monitoring of the platform. A new namespace, named "monitoring", is created in the cluster to handle all these pods that share a common monitoring functionality:

```
      $ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt␣
↪main.yml --tags "prometheus-deploy"
      $ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt␣
↪main.yml --tags "loki-deploy"
      $ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt␣
↪main.yml --tags "promtail-deploy"
      $ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt␣
```

(continues on next page)

(continued from previous page)

```
↪main.yml --tags "node-exporter-deploy"
$ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt main.
↪yml --tags "mongodb-exporter-deploy"
      $ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt␣
↪main.yml --tags "grafana-deploy"
      $ ansible-playbook -i inventory/<inventory name> --vault-id <inventory name>@prompt␣
↪main.yml --tags "homer-deploy"
```

**Congratulations!** These are all the required playbooks that need to be deployed to finish the Quobis wac installation. In the next section you'll learn how to check that everything is working properly.

### 4.1.6 Checking the status of the platform

Once all previous steps have been completed and all components have been installed, it is necessary to make sure that each individual process taking part in the deployment is running. You can see the namespaces created by issuing the following command. Please note that the actual output in your system might defer depending on the Quobis wac version that you are deploying:

```
$ kubectl get namespaces

NAME              STATUS   AGE
cert-manager      Active   90d
default           Active   91d
ingress-nginx     Active   91d
kube-node-lease   Active   91d
kube-public       Active   91d
kube-system       Active   91d
monitoring        Active   91d
velero            Active   34d
wac               Active   89d
```

The "default" namespace will be named accordingly to the "namespace" variable value that you have given in the Ansible main.yml configuration step.

On a kubernetes based deployment, these processes correspond with *pods* running inside the kubernetes cluster. Pods are the smallest, most basic deployable objects in Kubernetes. A pod represents a single instance of a running process in your cluster. Pods contain one or more containers, such as Docker containers. When a pod runs multiple containers, the containers are managed as a single entity and share the pod's resources.

To get the list of deployed pods, `kubectl` must be used. The kubectl command line tool lets you control a Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs. From a technical point of view, `kubectl` is a client for the Kubernetes REST API. In our case, just run the following command into a terminal and this will output something similar to this output below. Please note that the actual output in your system might be slightly different depending on the Quobis wac version that you are deploying, this specific one fits for v5.9.0:

```
$ kubectl get pods -A

NAMESPACE       NAME                                        READY   STATUS    RESTARTS␣
↪       AGE
cert-manager    cert-manager-99bb69456-kmdkc                1/1     Running   0       ␣
↪       31d
cert-manager    cert-manager-cainjector-ffb4747bb-rvmpp     1/1     Running   0       ␣
↪       31d
cert-manager    cert-manager-webhook-545bd5d7d8-w4qc2       1/1     Running   0       ␣
↪       31d
```

(continues on next page)

(continued from previous page)

```
default         dbgate-574bc5d58c-jb8h8                       1/1     Running   0          ␣
↪       31d
default         efs-provisioner-5b4fd7786c-d6nt6              1/1     Running   0          ␣
↪       33d
default         mailhog-8b5bfd58b-f7fsm                       1/1     Running   0          ␣
↪       31d
ingress-nginx   ingress-nginx-controller-6d846bd47b-zft79      1/1     Running   0          ␣
↪       36d
kube-system     aws-node-j6w57                                1/1     Running   0          ␣
↪       68d
kube-system     aws-node-qdh59                                1/1     Running   0          ␣
↪       30d
kube-system     coredns-6866f5c8b4-7r4dt                      1/1     Running   0          ␣
↪       31d
kube-system     coredns-6866f5c8b4-vgtv6                      1/1     Running   0          ␣
↪       31d
kube-system     ebs-csi-controller-97c95489b-8ms2k            6/6     Running   0          ␣
↪       31d
kube-system     ebs-csi-controller-97c95489b-k7bl8            6/6     Running   0          ␣
↪       31d
kube-system     ebs-csi-node-5b2zf                            3/3     Running   0          ␣
↪       68d
kube-system     ebs-csi-node-zsb9p                            3/3     Running   0          ␣
↪       30d
kube-system     efs-csi-controller-5b44d5b6b-bthbw            3/3     Running   0          ␣
↪       68d
kube-system     efs-csi-controller-5b44d5b6b-kc6r9            3/3     Running   0          ␣
↪       31d
kube-system     efs-csi-node-8wsdf                            3/3     Running   0          ␣
↪       30d
kube-system     efs-csi-node-zpz4n                            3/3     Running   0          ␣
↪       68d
kube-system     kube-proxy-7q4kp                              1/1     Running   0          ␣
↪       68d
kube-system     kube-proxy-tnzhn                              1/1     Running   0          ␣
↪       30d
monitoring      grafana-54456fb674-tfsmq                      1/1     Running   0          ␣
↪       27d
monitoring      kube-state-metrics-5fb588998d-tdmzf           1/1     Running   0          ␣
↪       33d
monitoring      loki-5fb6b44b79-wm49v                         1/1     Running   0          ␣
↪       32d
monitoring      metrics-server-6cb96b5c97-6t5h8               1/1     Running   0          ␣
↪       33d
monitoring      metrics-server-6cb96b5c97-gjkqz               1/1     Running   0          ␣
↪       33d
monitoring      node-exporter-px5v9                           1/1     Running   0          ␣
↪       30d
monitoring      node-exporter-w55d7                           1/1     Running   0          ␣
↪       33d
monitoring      prometheus-765c56fb54-pqtc5                   1/1     Running   1 (21d␣
↪ago)     21d
monitoring      promtail-ckv2g                                1/1     Running   0          ␣
↪       33d
monitoring      promtail-q9h5q                                1/1     Running   0          ␣
↪       30d
```

(continues on next page)

(continued from previous page)

```
velero          velero-56d599574f-sw6mf                       1/1     Running   0          34d
wac             audiomixer-sfu-1-0                            1/1     Running   1 (21h ago)   32d
wac             database-mongo-0                              2/2     Running   0          28d
wac             database-mongo-1                              2/2     Running   0          28d
wac             database-mongo-2                              2/2     Running   0          28d
wac             erebus-74bbf998c8-cq4vn                       1/1     Running   0          23d
wac             keycloak-64b8b6b5d8-btvmj                     1/1     Running   0          31d
wac             kitter-67d86469f8-djlxh                       1/1     Running   0          23d
wac             manager-6db5cc4c4-p2qvf                       1/1     Running   0          42d
wac             message-broker-7c958b46b4-dwp6v               1/1     Running   1 (5d21h ago)   28d
wac             mongodb-kubernetes-operator-86cf76bb88-2frz6  1/1     Running   0          31d
wac             postgresql-exporter-5f446765f8-ddt66          1/1     Running   0          31d
wac             postgresql-fcdf6cf9-xhdvw                     1/1     Running   0          31d
wac             qss-audiomixerio-sfu-1-5f9b4cc47d-5t5j7       1/1     Running   0          23d
wac             qss-audiostatus-6b847b74dd-9bq29              1/1     Running   0          23d
wac             qss-auth-http-74f455d976-ff54s                1/1     Running   0          23d
wac             qss-calls-588494879c-cf2gx                    1/1     Running   0          23d
wac             qss-calltransfer-basic-847bf96965-xxz9x       1/1     Running   0          23d
wac             qss-conference-state-f88f7f795-5srdq          1/1     Running   0          23d
wac             qss-invites-rooms-7f5f48865b-fflzr            1/1     Running   0          23d
wac             qss-io-websockets-6cf9c7fc94-w8wr9            1/1     Running   0          23d
wac             qss-log-conference-65964b85b6-bnv4f           1/1     Running   0          23d
wac             qss-meeting-basic-55b88b4469-mqn8z            1/1     Running   0          23d
wac             qss-peer-jt-855bbbbcdd-ms8mm                  1/1     Running   0          23d
wac             qss-quick-conference-6d565f958-zm22k          1/1     Running   0          23d
wac             qss-registry-authenticated-8648f484db-48spr   1/1     Running   1 (22d ago)   23d
wac             qss-resolver-wac-5cc787f85f-lcktg             1/1     Running   0          23d
wac             qss-rooms-basic-685f87f669-94tgd              1/1     Running   0          23d
```

(continues on next page)

(continued from previous page)

```
wac             qss-trunk-58777cf66b-mbpzs                    1/1     Running   0          ␣
↪        23d
wac             qss-watchdog-invites-7b96799fb5-b2phq         1/1     Running   0          ␣
↪        23d
wac             qss-watchdog-registry-79d8587fd4-tzphv        1/1     Running   0          ␣
↪        23d
wac             sfu-1-0                                       1/1     Running   5 (21h␣
↪ago)      32d
wac             sfu-dispatcher-9fc849b6c-5gfhr                1/1     Running   0          ␣
↪        23d
wac             sfu-wrapper-sfu-1-78689c7bf-k2256             1/1     Running   2 (22h␣
↪ago)      23d
wac             sippo-storage-54c9fd46d-fln24                 1/1     Running   0          ␣
↪        32d
wac             sippo-storage-54c9fd46d-sbsv2                 1/1     Running   0          ␣
↪        31d
wac             wac-core-786d7bff99-8ll9j                     1/1     Running   0          ␣
↪        12d
wac             wac-core-786d7bff99-g4wkf                     1/1     Running   0          ␣
↪        12d
wac             webphone-angular-9f5c6cc97-vt2fk              1/1     Running   0          ␣
↪        22h
wac             xmpp-server-cb78989d4-j52vd                   2/2     Running   27 (13d␣
↪ago)     28d
```

On this list you should be able to see at least one pod per role configured in the Ansible script and all pods should have the **STATUS** column set to *Running*. If some element is missing you should consider restarting the installation process making sure all steps have been followed correctly.

If one or more pods have a status different from *Running*, it means that those services have encountered an error. To to show details of a specific pod and related resources on a terminal, run the following command:

```
$ kubectl describe pod <name-of-the-pod>
```

The following is an example output for a describe of a `wac-core` pod:

```
Name:           wac-core-6b7c6f7684-lsxp5
Namespace:      poc
Priority:       0
Node:           ip-172-32-33-233.eu-west-1.compute.internal/172.32.33.233
Start Time:     Thu, 26 Dec 2019 11:13:21 +0100
Labels:         app=wac-core
                pod-template-hash=2637293240
Annotations:    <none>
Status:         Running
IP:             100.96.4.103
IPs:            <none>
Controlled By:  ReplicaSet/wac-core-6b7c6f7684
Containers:
wac-core:
    Container ID:   docker://
↪dddc63acfc156b252af7958601f6245bfd27f43f744ab003d792d5794b2ef015
    Image:          registry.quobis.com/quobis/wac-core:19.2.0
    Image ID:       docker-pullable://registry.quobis.com/quobis/wac-core@sha256:d22b6..
↪e664c372f4bd
    Ports:          8000/TCP, 5678/TCP
```

(continues on next page)

(continued from previous page)

```
    Host Ports:     0/TCP, 0/TCP
    State:          Running
    Started:      Thu, 26 Dec 2019 11:14:34 +0100
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
    /home/nfs/filesharing from kube-nfs-pvc-filesharing (rw)
    [...]

Conditions:
Type              Status
Initialized       True
Ready             True
ContainersReady   True
PodScheduled      True
Volumes:
wac-config:
    Type:      ConfigMap (a volume populated by a ConfigMap)
    Name:      wac-config
    Optional:  false
[...]

QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:          <none>
```

Also the following command can be executed to show the logs for the selected pod:

```
$ kubectl logs <name-of-the-pod> --tail 50 -f
```

An example of the output for this command is as follows (truncated to fit the page width):

```
Thu Jan 02 2020  /wac/lib/core/io/wapi/Wapi.js] debug: onWAPIMessage 877551, /sessions/
Thu Jan 02 2020  /wac/lib/core/io/wapi/Wapi.js] debug: onWAPIMessageResponse 877551, /
↪sessions/5e0e625ecc8296cb78101f79, PUT
Thu Jan 02 2020 /wac/lib/core/io/wapi/Wapi.js] debug: client disconnection with session␣
↪id 5e0e625ecc8296cb78101f79
Fri Jan 03 2020 /wac/lib/core/Sessions.js] debug: Remove session due expiration, [
↪'5e0e625ecc8296cb78101f79' ]
Fri Jan 03 2020 /wac/lib/services/UserSettings.js] debug: user4@acme.com disconnected
```

## 4.2 Upgrading from v4.4 to v5.0

A migration guide is available for customers willing to upload from previous versions to v5.0. Please contact your Quobis service manager for more information.

CHAPTER

FIVE

# USER MANAGEMENT

This section explains how users are managed in the platform, how the authentication process works, how they are classified into domains and *groups* and how to authorize them to perform different actions with *capabilities* and *permissions*

## 5.1 User management

### 5.1.1 User roles

There are three types of user's roles in Quobis wac:

- **admin**: Used for reporting roles and management. These users will have access to all the content on the system and will not be limited by permissions and/or capabilities
- **user**: for standard uses, they are limited by user/domain capabilities.
- **anonymous**: intended for external users with a limited scope of time or a single session

Users on the system are organized on *Domains*. The domains allow to spread capabilities and provide services on a limited way to a selected group of users. Different use cases could take advantage of this concept, like different companies sharing the same instance, or maybe different offices of the same company. Every resource on the system belongs to a domain, so there is a default domain configured on the system. There is a base domain called `quobis` but extra domains can be created based on your use case requirements.

### 5.1.2 User IDs

Users have a unique identifier, the *UUID*, which is tied to the database ID. It can refer to a user, a group or to any other entity that can be referred on the system. This is used to resolve usernames, alias and any other element user-friendly to refer to a functional element.

### 5.1.3 Usernames

In addition to the UUID, users are also identified by its username. There is no restrictions on the minimum of maxinum number of characters of a username.

Please note that usernames in the wac-core service are unique per domain, whereas usernames in Keycloak are unique per realm. For the sake of simplicity, we use a single realm in Keycloak which leads to restrict, in practice, to have unique usernames at WAC level.

In addition, we sometimes refer in the documentation to "username" as the combination of username and domain.

### 5.1.4 WAC-URIs

In order to route calls, the wac-core and QSS service understand three different types of entities:

- *wac-user:UUID*: this entity refers to a user identified by its UUID
- *wac-group:UUID*: this entity refers to a user group, identified by its UUID
- *wac-voicemail:UUID*: this entity refers to a voicemail system, and the UUID is the UUID of the user in that system

Quobis SDK, Quobis WAC and QSS service use these three identifiers to route calls, notify users, etc... In some cases, users are also identified by its username and domain.

## 5.2 Authentication

Authentication is performed using OpenID Connect protocol. For more information about the details of OpenID Connect please refer to the official specification.

Keycloak (an OpenID Provider) is included in the product to allow authenticating to all the client applications with that need. Keycloak is an open source Identity and Access Management solution that works as OpenID Provider. An OpenID Provider is an OAuth 2.0 Authorization Server that is capable of Authenticating the End-User and providing Claims to a Relying Party about the Authentication event and the End-User. For more information about Keycloak please refer to the official documentation.

### 5.2.1 Obtaining an access token

For testing, development or administration purposes we may need to obtain an access token to be able to access the REST APIs. We can obtain one using the Direct Access Grants flow. This means that client has access to username/password of user and exchange it directly with Keycloak server for an access token. In terms of OAuth2 specification, this is called "Resource Owner Password Credentials Grant".

```
POST https://<keycloak-host>/auth/realms/<realm>/protocol/openid-connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
{
    grant_type: "password",
    client_id: <client-id>,
    username: <username>,
    password: <password>,
}
```

Please note that you need to know both the username, password and client-id for this user. The conneciton can be achieved for example using a command-line curl with the following command:

```
curl -X POST https://<keycloak-host>/auth/realms/<realm>/protocol/openid-connect/token -
↪d "grant_type=password&client_id=<client-id>&username=<username>&password=<password>"

{
    "access_token":
↪"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJaTzQ5NFN1QjNBdThMcjMwVTRsQVoxdGNHVXZ2eXhYalA4VG5kNz
↪",
    "expires_in": 4200,
    "refresh_expires_in": 864000,
    "refresh_token": "eyJhbGciOiJy0BD7p9K6A-3Q11Yt3RLg",
    "token_type": "Bearer",
    "not-before-policy": 1647863488,
    "session_state": "737cb7df-4d2f-41ca-a25a-e4066d8d9ef3",
```

(continues on next page)

(continued from previous page)

```
    "scope": "email profile"
}
```

And we can see the response with the requested access_token `eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSld.....` In order to use the access token, you just need to include it in the Authorization header like in the next example:

```
curl -X GET https://rest-api-IP-of-your-system/sapi/users -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJaTzQ5NFN1QjNBdThMcjMwVTRsQVoxdGNHVXZ2eXhYalA4VG5kNzV"
```

### 5.2.2 Keycloak configuration

- When adding a SAML v2 Identity Provider, please make sure that the parameter "Force Authentication" is set to "OFF" in order to properly logout. More information can be found in the Keycloack documentation site.

#### 5.2.2.1 User session count limit

Keycloak's last versions added a new variable that allows the system to limit how many sessions a user can create. In order to add this configuration, the Authentication Flows must be modified (Keycloak recommends, at least, Browser, Direct Grant and Reset Credentials).

The configuration for these limits is as follows:

```
{
"id": "92d5ac54-f1e6-4cef-907f-22f011526500",
"alias": "Sessionslimit",
"config": {
  "userClientLimit": "0",
  "behavior": "Deny new session",
  "userRealmLimit": "3"
}
```

It is required to add an alias, so it can be called in the different Flows. Limits and behavior can be configurable. Limits can be applied per client or per realm.

First one (userClientLimit) limits the maximum number of sessions a user can have with each client while the second one (userRealmLimit) limits the number of sessions per user in the whole realm

The behaviour defines how the limits work. The are two options:

- Deny new session. If there are a number of sessions equals to the limit, it will deny the access to the new session.
- Terminate oldest session. If the are a number of sessions equals to the limit, it will delete the oldest one an create another one session.

It is remarkable that a deleted session is not a logout. The session will be alive until the window or application are closed.

### 5.2.3 Automatic discovery of identity providers

By default, Keycloak presents a username/password form and a list of configured identity providers to the user. When a federated user wants to login via Keycloak, the user needs to choose an identity provider to get redirected. These instructions explain how to configure Keycloak to allow skipping the step of selecting an identity provider.

The idea is modifying the browser based login flow so Keycloak presents a username only form (i.e.: without password form and without the list of configured identity providers). User can then enter their email address and Keycloak will choose an identity provider based on the provided email domain and forward the user to the chosen provider.

Details about how to configure this functionality can be found here: https://github.com/sventorben/keycloak-home-idp-discovery/tree/v20.0.1#how-to-configure. A summary of the required steps is explained below:

- Go to Authentication, Flows. Duplicate browser based login flow.
- In the just created flow, add a new step "Home IdP Discovery" just before the part of the standard username and password form.
- Bind the just created flow as "Browser flow".
- Associate the email domains with each ID provider:
  - Enter into the Keycloak container
  - Configure kcadm:

```
export PATH=$PATH:/opt/keycloak/bin
kcadm.sh config credentials --server http://localhost:8080 --realm master --
↪user admin --client admin-cli
```

  - Associate ID provider to a domain

```
kcadm.sh update realms/<realm>/identity-provider/instances/<idp name> -s
↪'config."home.idp.discovery.domains"="example.com##example.net"'
```

  - Check out that it was correctly added:

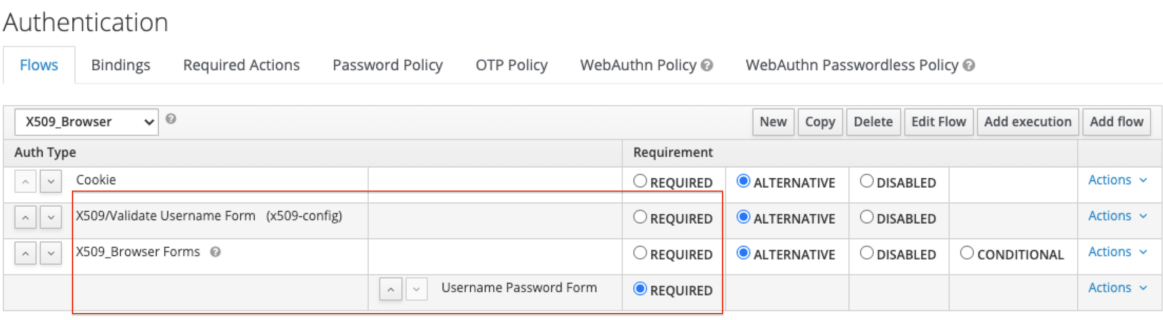
```
kcadm.sh get realms/<realm>/identity-provider/instances/<idp name>
```

### 5.2.4 Single sign-on with personal certificates (X.509 Client Certificate Authentication)

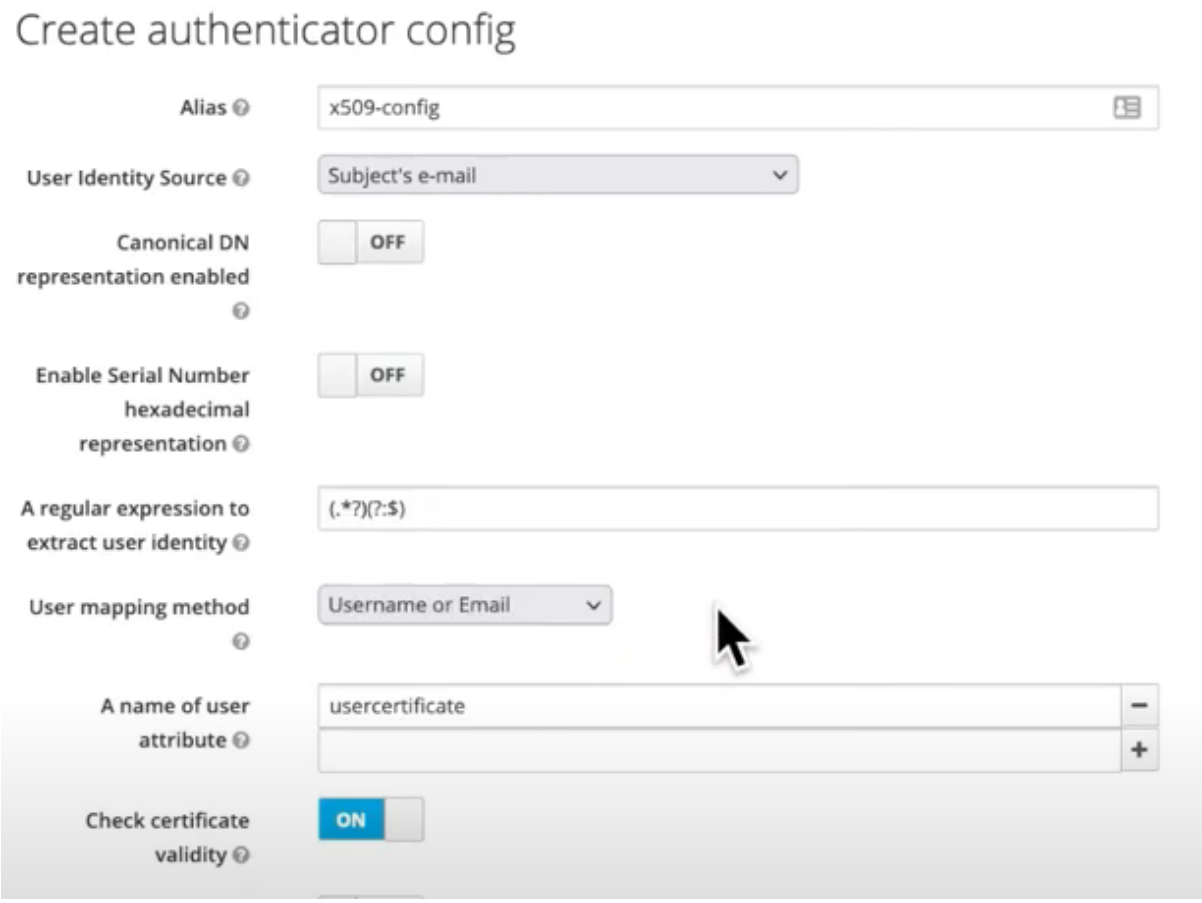
A personal digital certificate, also known as "digital certificate" or Citizen's Certificate, is a electronic certification issued by a public Certificate Authority that links the holder to a set of signature verification data and confirm his/her identity. It enables users to identify themsevles online and share information with other people and bodies, safe in the knowledge that only him and his/her interlocutor are able to access that information.

Quobis WAC can be configured to authenticate users using a digital certificate using the well-known X.509 standard. This is achieved by following these steps:

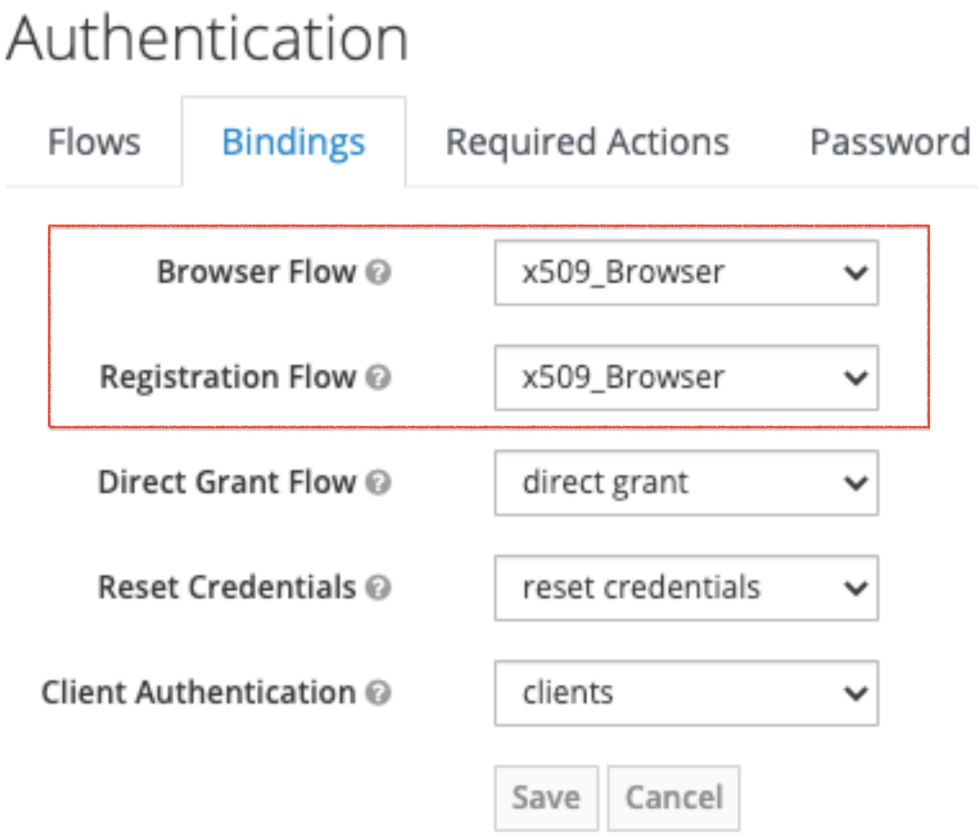
- In the Keycloak "Authentication" section, you need to add a new authentication type as shown below. Please make sure that it's set as "Required":




- A new "authentication flow" needs to be created. Select "certificate serial number" or any other atribute as the user identity source. The certificate identity mapping can be configured to map the extracted user identity to an existing user's username or e-mail or to a custom attribute which value matches the certificate identity. For example, setting the Identity source to Subject's e-mail and User mapping method to Username or email will have the X.509 client certificate authenticator use the e-mail attribute in the certificate's Subject DN as a search criteria to look up an existing user by username or by e-mail. There are other optional features available here such as the extended certificate validation or to check the revocation status with different methods.

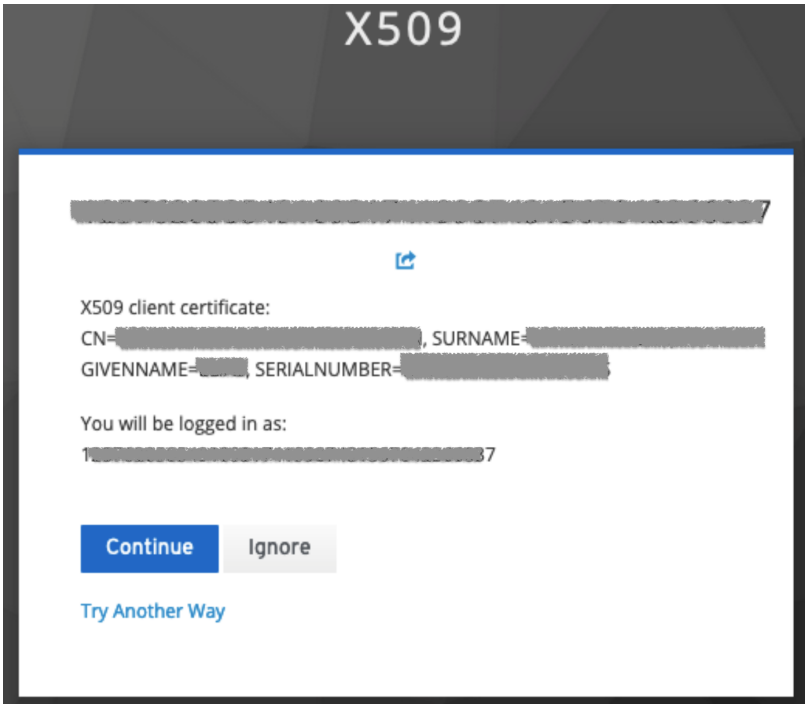



- This new authentication flow that you have just created needs to be added in the Bindings section as shown below:

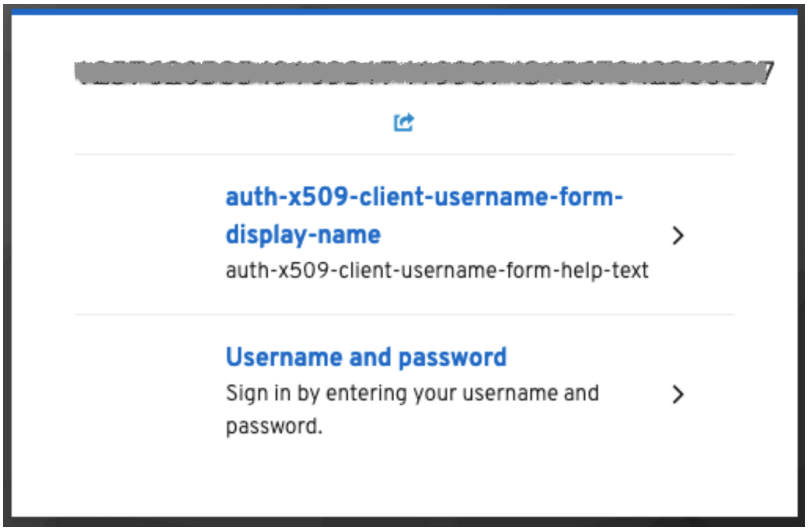



- Once this is configured, new users will see a pop-up window where the browser will inform the user that a user certificate will be used and shared with the server for authentication (some information has been removed for privacy reasons):

- After that pop-up window, Keycloak will specify again the details of the digital certificate and the user can either accept or fallback to another authentication mechanism by clicking on “Try another way”




- This is the fallback window shown when the user chooses to login with another method:




More information can be found in the Keycloak documentation: https://www.keycloak.org/docs/latest/server_admin/#_browser_flow

### 5.2.5 Single sign-on (SSO) with Google

Keycloak can be configured to delegate authentication against Google identity services to implement a Single sing-on service (SSO). This chapter describes the configuration needed both in the Google console and the Keycloak server.

#### 5.2.5.1 Google console configuration

- Create a project in console.cloud.google.com.
- Go to "OAuth consent screen" and select which users are allowed to login:
  - Internal: Users accounts from our organization
  - External: Every Google account
- Fill the following fields: App name, User support email, Authorized domains & Email addresses
- Go to "Credentials" and Create Credentials with OAuth client ID
- Use "Web application" as Aplication type and fill its name.
- Copy the "Authorized redirect URI" from our Keycloak console→ Identity Provider→ Google → Redirect URI

#### 5.2.5.2 Keycloak

- Select "Google" from the "Identity Providers" section in the realm that you want to use for this authentication scheme.
- Get the ID and Secret from the credentials created before.
- Go to "Authentication" and Create Flow with "Detect existing broker user" and "Automatically set existing user"
- Configure Google in "Identity Provider" to use this Flow as "First Login Flow".

## 5.3 Domains

### 5.3.1 Domains, routing and user visibility

The domain is the main framework or container of the users. Any user needs to be assigned to a domain.

Domains are nested in parent-child relationship, starting from a domain named by default as "quobis". **This is the top-level domain and cannot be changed, renamed or removed**. In addition, the "quobis" domain is the default one when no domain is given when mapping users to domains. Every domain need to have one and only one parent domain, in a nested parent-child relationship as shown belown:

```
- quobis (top-level and default domain)
     |___ anonymous (only since v5.0.0)
     |___ domain1
     |___ domain2
             |___ domain3
             |___ domain4
     |___ domain5
             |___ domain6
                     |___  domain7
                     |___  domain8
```

Since version v5.0.0, there is also a domain named "anonymous" that is automatically created by default and that cannot be edited or removed. All anonymous users are mapped into this domain, which is a "child" of the main "quobis" domain.

The users of a domain have all the other users of the domain accessible to reach for presence and other information. Users from other domains are still reachable (you can dial and call them, using the `username@domain` unique identifier), but not directly accessible as other users from your own domain.

Any request on your application that involves an username without specifing the domain will result on the resolution along your own domain. For example, in the case that there are two users on the system, namely `bob@quobis` and `bob@acme`, if you as user of the `quobis` domain (`alice@quobis`) try to call `bob`, it will be resolved as `bob@quobis`. You always have the option to call "the other Bob" if you dial directly to `bob@acme`.

### 5.3.2 Listing domains

An administrator user can retrieve the list of the domains on the system and their properties;

```
GET https://wac:8001/sapi/domains HTTP/1.1
Authorization: Bearer ec3fe973ea047b6fd38228eb9f9b9661cbb5b0fdeac01ff13af1ee
Content-Type: application/json

[
    {
        "id": "57877673e8f15d3472817349",
        "parent": null,
        "domain": "quobis",
        "origins": [null],
        "enableAnonymous": false,
        "anonymousExpiration": 0,
        "anonymousCapabilities": [],
        "services": {
            "callalarms": [],
            "callcontrol": [],
            "chat": [],
            "contacts": [
                "wac",
                "static"
            ],
            "datapipe": [],
            "filetransfer": [],
            "forms": [],
            "im": [],
            "ldap": [],
            "login": [],
            "meetings": [],
            "oauth2client": [],
            "presence": [],
            "remotelog": [],
            "usersettings": ["wac"]
        }
    }
]
```

Where the following fields can be found:

Table 1: Domain values

| Parameter | Value |
|---|---|
| id | UUID of the domain. To be used for other API operations. |
| parent | This field will indicate the parent domain. Parent domain limitations will be extended to the the child domain. |
| domain | Name of the domain |
| origins | Used to map the domain specified on the `username` login body request with the anonymous credential assigned. |
| enableAnonymous | Anonymous login enabling option. |
| anonymousExpiration | Time in milliseconds anonymous users are allowed to perform operations on the system before they will be requested to refresh the authentication. |
| anonymousCapabilities | List of capabilities of the anonymous users for this domain. Check *Users capabilities* to expand information about this topic. |
| services | List of services available on the system. Some ones will include extra details in order to define the backends included on it. |

**Getting details about an specific domain**

Once you get the UUID of the domain, you can retrieve the details of your domain. The READ access to the user's domain details is granted for all users of the domain.

```
GET https://wac:8001/sapi/domains/57877673e8f15d3472817349 HTTP/1.1
Authorization: Bearer ec3fe973ea047b6fd38228eb9f9b9661cbb5b0fdeac01ff13af1ee
Content-Type: application/json
```

### 5.3.3 Creating a domain

On multi-tenant environments or based on the specific business needs, it is recommened to have separated domains for different sets of users. This can be easily managed using the REST API to create the desired domains for the system.

```
POST https://wac:8001/sapi/domains HTTP/1.1
Authorization: Bearer ec3fe973ea047b6fd38228eb9f9b9661cbb5b0fdeac01ff13af1ee
Content-Type: application/json

{
    "domain": "demo.quobis.com",
    "parent": "quobis",
    "origins": ["demo.quobis.*"],
    "enableAnonymous": true,
    "services": {
        "login": [],
        "presence": [],
        "callcontrol": [],
        "contacts": ["wac", "static"]
    }
}
```

The previous example will create a new domain on the system that enables anonymous access and limit the services available to these users to the specified ones.

As you can understand it is specially interesting on use cases where anonymous users are involved. But also allows to create several domains where the users are not visible between them.

### 5.3.4 Update a domain

Typical use cases to update a domain is to grant a new service, remove it or to assign a new domain for the anonymous REFERRER header.

On this case, just use the PUT command with the UUID of the domain.

```
PUT https://wac:8001/sapi/domains/578895afe8f15d347281734f HTTP/1.1
Authorization: Bearer ec3fe973ea047b6fd38228eb9f9b9661cbb5b0fdeac01ff13af1ee
Content-Type: application/json

{
    "domain": "demo.quobis.com",
    "parent": "quobis",
    "origins": ["demo.quobis.com", "showcase.quobis.com"],
    "enableAnonymous": false,
    "services": {
        "login": [],
        "presence": [],
        "callcontrol": [],
        "contacts": ["wac", "static"]
    }
}
```

## 5.4 User Groups

### 5.4.1 Introduction

In addition to the classification of users into domains, Quobis WAC allows the creation of group of users that have something in common or share some characteristics. This is a quite common practice for example in business environments, where users are classified into departments or teams, but it also applies to a lot of other use cases where we need to combine users into different sets. Quobis WAC provides a number of functionalities to ease the management of these groups and to automatically include group calling and group chatting features.

In Quobis WAC, an "user group" is a set of system users (also referred to as "users group participants") that has the following characteristics:

- It is given a name (a string)
- It has an associated chat group
- It is a "callable" entity
- It has one or more associated *phonebooks*

The associated chat group, which has the same name as its associated user group, is automatically synchronized with the group participants. That means that if the owner adds/changes/removes an user from the userGroup, the system will automatically add/change/remove that user from the ChatGroup. On the other hand, when a user is added to a user group by someone else, the system will include that chat group in the list of groups that this user belongs to.

## 5.4.2 Calling to an user group

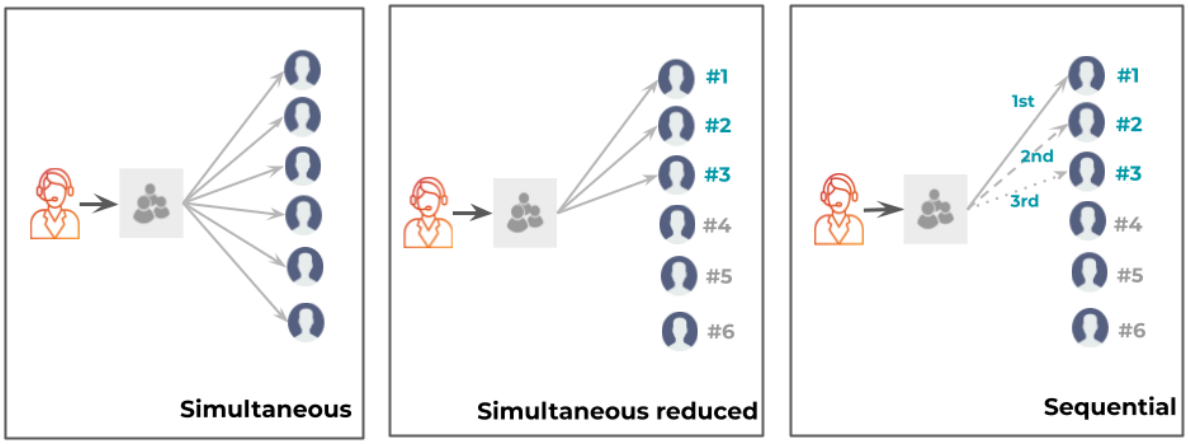
The fact that groups are "callable" means that a audio or video call be placed to them. A user group can also be a destination of a call transfer (attended or direct). In order to do that, the ID that needs to be called when making the call is the user group ID.

Please note that this behaviour means that the callee does not make a call to each group participant, it's the signaling server which makes that on its behalf. That also means that the caller does not know the actual users that he/she is calling when dialing into the user group. This call is registered in the call details records as a regular call, being the callee the user group ID.

This behaviour is also referred by some PBX vendors as a "ringing group", which in our case is just a feature of the user groups.

### 5.4.2.1 Calling strategies for ringing groups

The user groups can be configured so the incoming calls are routed to the participants by following different calling strategies. It is important to clarify that the participants of an user group follow an order, according to its position in the database (as stored in JSON format in the "participants" field in the /usersGroup/{groupId} endpoint). This order is taken into account when calling to an user group in some calling strategies, as will be explained below.




#### Simultaneous

When this strategy is selected for a user group, incoming calls to this user groups will be routed automatically to every user group participants. That means that the call will ring for every participant at the same time. The first participant who takes call will be connected to the caller and the call will stop ringig for the rest of participants.

#### Simultaneous reduced

When this ringing strategy is selected, only the first "maxSteps" participants will be called, all of them at the same time. The rest of the participants won't receive incoming calls at all. This is useful when a user group is large and we don't want to call every participant.

### Sequential

When this ringing strategy is selected, the system will start calling the participants of the user group, one after another, following the order in the group unntil the participant "maxSteps" is reached. In addition, the delay between calling a participant and calling the next one can be configured (parameter "stepDelay"). This behaviour is set according to these two parameters of the QSS:

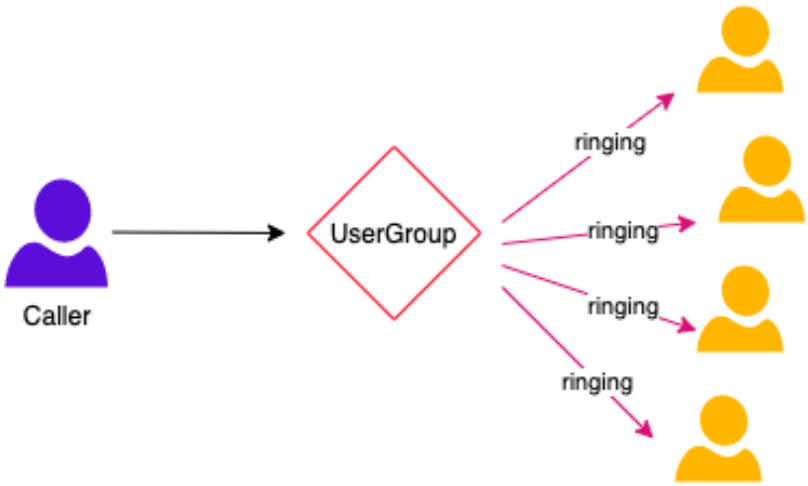
```
"ringingGroup": {
            "stepDelay": 0,
            "maxSteps": 5
    },
```

This is the effect of these two parameters:

- Parameter **stepDelay**: defines the number of seconds to wait before calling the next participant in the list. Defaults to 0 seconds, maximum value 60 seconds.
- Parameter **maxSteps**: defines the maximum number of participants that will be called. For example, if we have a usergroup with 10 participants and the value of maxSteps is 5, then only the fist 5 participants will be called. This parameter is only taken into account when te value of stepDelays is different from zero. Its default value is five participants.

This functionality has been implemented as follows:

- If "stepDelay" is set to zero, the calling strategy is like a simultaneous ringing.
- If a user answers the call, it will be connected to the caller party. The remaining calls will stop ringing inmediately.
- If an user rejects the call, that action won't be notified to the caller party and the call will keep ringing for the rest of participants.
- If no one accepts the call (not answering or rejecting) the caller will receive a "No answer" message or "Busy" messages (depending on the answer of the last called participant)



### 5.4.3 Known issues

Other characteristics of the user groups are:

- User groups have a owner, which is a participant who can add, change or remove users from the usersGroup that he/she has created.
- A user group can only have one owner and it cannot be changed after it's created.
- A participant can be owner of one of more user groups.
- A participant can belong to one of more user groups.
- There is no limit in the number of users in a UserGroup
- The associated chat group is not a standard one so no actions can be done via the REST API with these chats. The system will remove it automatically only after the user group is removed. Attempting to interact with these type of chat groups via the REST API will thrown an error.
- In the event that a user is transferring or calling to a group to which he/she belongs to, he/she won't be included in the callee list. For example, if Alice belongs to a user group with Bob, Carol and David, and Alice calls into that user group, only Bob, Carol and David will get an incoming call.
- An incoming call to a user group will end when the global call timeout is reached (as set in the configuration variable "watchdog.invites.expiration" of the QSS.watchdog-invites service) even if the "maxSteps" number of participants has not been reached.
- Multi-device environments: please note that, in case that a user has active sessions in several devices, each device will be considered as a participant. That means that, in the case of sequential ringing, the call will ring first on one device, then on his second device, etc... instead of jumping to the next participant. This behaviour may be fixed in future versions".

### 5.4.4 Configuration

The service needs to be enabled in the wac.ini file in order to work, with the following configuration (it is activated by default after a standard deployment). There is no specific configuration rather than activating the service as shown below.

```
[usersgroup]
backend[] = UsersGroup

[backend.UsersGroup]
module = UsersGroup
```

This service can be reached via the *Quobis WAC REST APIs* where a number of methods are available to create a group, get a list of groups and other CRUD operations.

### 5.4.5 Log messages

These are the log messages associated with the user groups management:

- `error: Specified participant does not exist`: an error occurred when trying to add a participant that doesn't exist yet
- `error: User ${user.id} is not the owner of the group`: an error occurred when a user tries to manage a group in which he is not the owner.

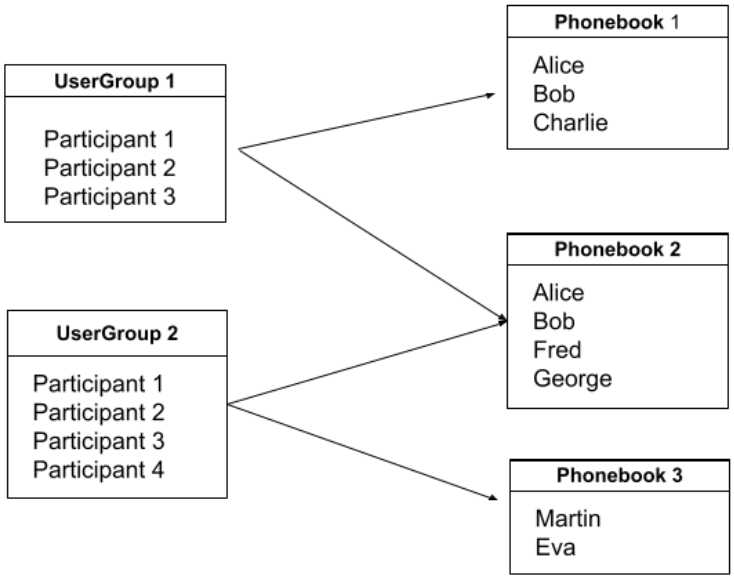
## 5.5 Phone books

### 5.5.1 Overview

A phonebook is a set of pairs of a name and an callable address, referred to by a name ("phone book name"), that can be associated to user groups. The phonebook service is commonly used to create lists of shared phone contacts as part of the agenda of the users that belong to the same user group.

Phone books are closely coupled with the *user groups* as user groups can subscribe to one or more phone books. There is no limitation neither in the number of contacts allowed in a phonebook nor in the number of userGroups subscribed to a given phonebook.

The following image explains with an example the relationship between user groups, user group participants and phone books. As seen in the image, the usergroup #1 is subscribed to phone books #1 and #2 while the usergroup #2 is subscribed to phone books #2 and #3.




### 5.5.2 Data structure

The name in an entry of a phonebook identifies the contact while the address can be regular phone numbers (in E.164 format), a system username (for example, "peter@acme.com") or event a user group "groupID" (as usergroups are "callable entities"). As a result, a phone book associated to a user group can have entries to call another user groups. There can only be one address entry per name. The data representation of a phone book is as follows:

Table 2: Data structure of a phone book

| Property | Type | Description | Required | Default |
|---|---|---|---|---|
| id | string | UUID of this phonebook. Read-only, generated by the system. | Yes | N/A |
| name | string | Name given to this phone book. | Yes | N/A |
| contacts[] | nested object | Contacts that make part of this phonebook | Yes | N/A |
| contacts[].na | string | Name of the contact | Yes | N/A |
| contacts[].ac | string | Address of the contact (regular phone number, system username or usergroup ID) | Yes | N/A |
| contacts[].id | string | UUID of this contact | Yes | N/A |
| subscribers | list | Array of strings with the ID of the user groups that are subscribed to this phone book | Yes | [] |

**Note:** Please note that users does not subscribe directly to a phone book. User groups are the ones who are subscribed to a phone book.

An example of this structure in JSON format is shown below, where the first entry is a system user or usergroup (102@quobis.com) and the second entry is a regular phone number.

Listing 1: Example of the structure of a phone book

```
{
 "id": "bb5e8a13d428d2ebce767b8",
 "name": "Useful phonebook",
 "contacts": [
     {
        "name": "Reception",
        "address": "102@quobis.com",
        "id": "bb5e8a13d428d2ebce767b8"
     },
     {
       "name": "Post Office",
       "address": "+34 902 999 222",
       "id": "a34e8a13d428d2ebce767b8"
     }
  ],
 "subscribers":["iduserGroup1", "iduserGroup1"]
}
```

### 5.5.3 Configuration

The phonebook service is configured via the wac.ini file. It contains a top-level entry named [phonebook] that holds the service's details.

Listing 2: Example of wac.ini entry to activate the phonebook service

```
[phonebook]
```

This service can be reached via the *Quobis WAC REST APIs* where a number of methods are available to create a phonebook, get a list of phonebooks and other CRUD operations.

## 5.6 Presence

### 5.6.1 Introduction

Quobis wac provides a presence functionality that allows users to know the "status" or presence of another user, and also setting its own presence. In addition to the "online" status, other information is stored in the presence as the user avatar, the displany name, a note or the "mood". This functionality is implemented in the wac-core service.

Presence is managed via a set of REST API endpoints, while its updates are sent via the websocket connection to the user subscribed to a given presence. Before getting updates, the user needs to "subscribe" to the presence of a given user. After that, he/she will receive updates everytime that the presence changes.

### 5.6.2 Presence and call management

The presence service is tightly coupled with the QSS service, as the presence of the user is changed to "busy" when she/he is on a call. On the other hand, the user is announced as "online" when she has a open connection with the server.

### 5.6.3 Known issues

For compatibility reasons, since v5.3, the presence information is not removed when deleting users, as we asume all users have a presence so if we drop presence for deleted users, applications querying data about those users may fail. Instead, when a presence query about a deleted user is done, the answer is an empty presence (it only contains the address field, the other fields are null or empty strings).

## 5.7 Users capabilities

### 5.7.1 Introduction

The list of available features or behaviors of a Quobis SDK application is determined by the resulting capabilities of the user session. Right after the user makes its first successful login, a set of **capabilities** are established. There are three types of capabilities:

- **Service capabilities**: defined by the `sippo-server` configuration and enabled services.
- **Stack capabilities**: defined by the vendor of the signaling stack.
- **Session capabilities**: defined by the wac-user or wac-domain configuration, where:
  - **Pruned capabilities**: defined by configuration to remove specific features per user or domain.
  - **Forced capabilities**: defined by configuration to add specific features per user or domain.

The applications will receive a set of capabilities following these conditions:

- All capabilities are provided by active services (system functionalities).

- All capabilities are limited by the stack (client browser or app limitations).
- Administrators can add capabilities for extra features on the client side.
- Administrators can limit any present capability.

Every user is able to retrieve their platform capabilities. The applications must adapt the user interface to the resulting status of the abilities granted. So, as a result, removing a capability, results on limit access to a feature.

```
GET https://wac:8001/sapi/capabilities/ HTTP/1.1
Authorization: Bearer ec3fe973ea047b6fd38228eb9f9b9661cbb5b0fdeac01ff13af1eef8

{
  "domain": [
    "stats",
    "alarms",
    "callcontrol",
    "chat",
    "contacts",
    "datapipe",
    "filetransfer",
    "login",
    "meetings",
    "login-token",
    "presence",
    "log",
    "settings",
    "recording"
  ],
  "user": [
    "[+c2cagent, +caller-extra-info]"
  ]
}
```

### 5.7.2 Service and stack capabilities

These capabilities define the available features for a user on an application:

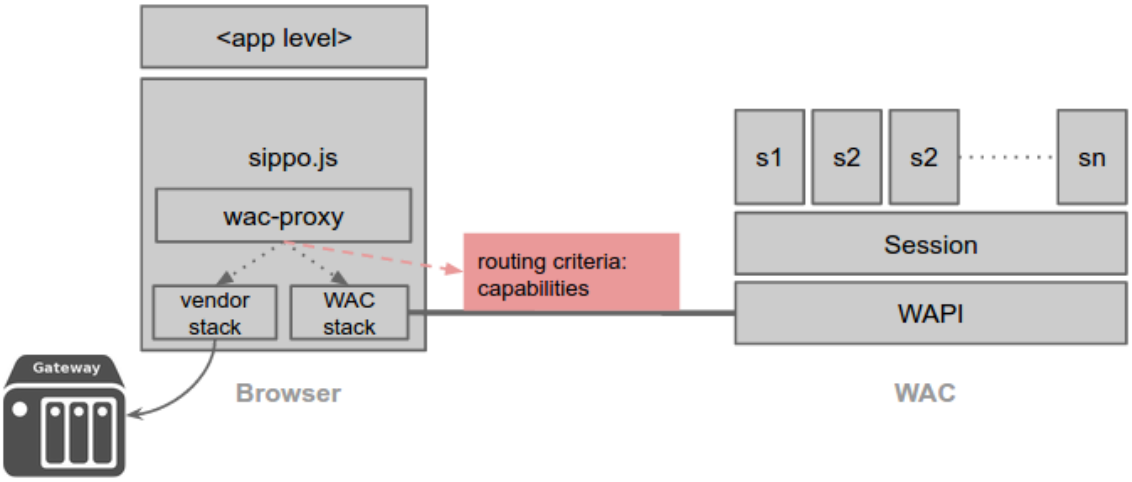
- **Service capabilities**: defined by the `sippo-server` configuration and enabled services.
- **Stack capabilities**: defined by the vendor of the signaling stack.

Depending on the configuration of the `sippo-server` and the stack used in each deployment, the set of available features in a Sippo environment will be different. The running services at the `sippo-server` determinate the functionalities available in the server side and provided by the `REST API`.

In the same way, some vendor stacks provide functionalities that are not available in others (like data channel or call transfer), maybe due to what the manufacturer uses as Media Server or due to platform limitations.

In the end, the set of available features is exposed to the applications on top of SippoSDK. The final application can use this to adapt the user interface (for example, do not show a call transfer button is the underlying stack does not support it), or for any other purpose (configuration or monitoring).

Some functionalities can be provided by the Sippo AS, others by the stack and others are provided by both elements. In the latter case the Sippo AS has a higher priority. The next picture represents how is the routing process. Whenever an application invokes a method on the SippoSDK, it is routed to the vendor stack or to the `sippo-server`. This decision is made in the `wac-proxy`, one of the internals modules of SippoSDK.




For configuration purposes. To generate the `session capabilities` the AS needs to know what services are running and create an array with the corresponding capability of each one. This is accomplished by the service `Capabilities`:

```
(wac.ini)
...
[capabilities]
...
```

### 5.7.3 Configurable session capabilities

In addition to the **Service capabilities** and the **Stack capabilities**, there are capabilities defined by configuration:

- **Pruned capabilities**: defined by configuration to remove specific features per user or domain.
- **Forced capabilities**: defined by configuration to add specific features per user or domain.

These are very useful to customize the behavior of some users or domains without affecting the other ones.

Forced capabilities are just strings for the `sippo-server`, that will be exposed to the applications without changes. An administrator is free to add extra configuration capabilities needed by the application. This way, any application can define new capabilities.

These capabilities are configured during the user provisioning process, by adding an array of capabilities to the user.

- Include a + as prefix to force (add) a capability to a user.
- Include a - as prefix to prune (remove) a capability to a user.

The following example will edit a user to include an extra capability (`caller-extra-info`) and to remove another one (`chat`).

```
PUT https://wac:8001/sapi/users/5788b1a8e8f15d3472817350 HTTP/1.1
Authorization: Bearer ec3fe973ea047b6fd38228eb9f9b9661cbb5b0fdeac01ff13af1eef
Content-Type: application/json

{
   "domain": "quobis",
```

(continues on next page)

(continued from previous page)

```
  "username": "alice",
  "email": "imalice@demo.quobis.com",
  "role": "user",
  "capabilities": [+caller-extra-info, -chat],
  "mobilePhone": ['1111111'],
  "alias": "masterOfTheUniverse"
}
```

**Note:**

- The update command must include all the fields that you require to add to the user, not just the ones to update, so `username`, `domain`, etc. must be included on each `update` operation.
- Every application defines a set of specific capabilities that could be used by the application. Check the specific application documentation for a complete list of available forced capabilities.
- A "+" or "-" sign needs to be added to the capability in order to take effect. Otherwise, adding just the capability name without the sign will have no effect at all.

### 5.7.4 Session capabilities

Complete list of Quobis wac capabilities. Each application will use them in order to provide more or less actions to their users.

#### 5.7.4.1 Configuration options

**call-from-chat**
Ability to call directly from the chat section.

**call-rehydration**
Ability to recover an existing call when refreshing page.

**caller-extra-info**
Ability to access to extra context information about an incoming call.

**change-camera**
Ability to change the camera during call.

**create-contacts**
Ability to create contacts.

**hide-incoming-domain**
Ability to display or hide the domain of the caller substituting it for the configured one.

**make-calls**
Ability to make calls.

**settings**
Ability to show a form in the client applications with some configurable parameters of the WAC or the core network.

**view-call-history**
Ability to display call history.

**view-contacts**
Ability to display contacts.

**view-meetings**
Ability to display meetings.

**whiteboard**
Ability to create a whiteboard session.

#### 5.7.4.2 Service capabilities

*Capabilities implemented based on services running on the ``sippo-server``. These will be active if the service is correctly configured and running in the back-end system.*

**call-alarms**
Ability to process received WebRTC stats data from the stack (stats-reporting) check defined alarms and trigger it if they match.

**call-history**
Ability to access to the call history endpoint.

**callcontrol**
Call routing and creation of one DB entry per call.

**chat**
Ability to create a chat (proprietary protocol) - If service present at sippo-server it prevails.

**contacts**
Ability to access to contacts endpoint.

**credentials**
The WAC acts as a storage service for credentials.

**datapipe**
Ability to create RAW data channels (based on Websocket, not WebRTC DataChannels) using Quobis AS as proxy.

**filetransfer**
Ability to send files using WAC as proxy. Chat capability is required.

**forms**
Ability to create user/agent forms to be filled by both.

**log**
The SDs sends local logs to the WAC for debugging purposes. Disabled by default.

**login**
Ability to login in the WAC with basic authentication (username, password).

**login-token**
Ability to login into the WAC by using a token-based mechanism, like OAuth2, which allows to use a third party authentication server.

**meetings**
Ability to view / create / remove meetings.

**presence**
Ability to receive presence updates about my contacts.

**recording**
Ability to record a call (infrastructure mode).

**sessions**
Sippo wac-session service is active on the Sippo AS.

**stats**
Ability to process received WebRTC stats data from the stack (stats-reporting) and send it to the Sippo AS for processing.

#### 5.7.4.3 Stack relying capabilities

*Capabilities implemented at SippoSDK level on the specific media integration. We can face limitations on this section based on the vendor running on your environment.*

**attended-transfer**
Ability to transfer a call to a third user from the on-hold panel.

**audio-call**
Ability to create an audio only call.

**audio-video-call**
Ability to create a combined audio & video channel.

**blind-transfer**
Ability to transfer a call directly to a third party user.

**collaboration-call**
Ability to create a call without audio or video.

**conference-hold**
Ability to hold a call when using a conference.

**conferences**
Ability to create multi-endpoint calls.

**datachannel**
Ability to create ciphered RAW data channels end to end.

**geturi**
Ability to obtain a URI assigned by the gateway (`ims` credentials).

**hold**
Ability to put a call on-hold.

**im**
Ability to create a chat using the IM caps of the stack.

**local-recording**
Ability to record a call directly on the user's browser.

**media-update**
Ability to manipulate and change the current audio / video streams.

**multicall**
Ability to have multiple 1-to-1 simultaneous calls.

**network-reporting**
Ability to receive periodic network reports checking the call quality on real-time.

**screen-sharing**
Ability to share screen and use it as video stream.

**stats-reporting**
The underlying stacks reports statistics about the current call.

**video-call**
Ability to create a video channel.

## 5.8 Permissions

### 5.8.1 Introduction

Permissions are a new way of managing the way that administrators and developers can set up domain-wide and per-user policies. This feature has been introduced in v4.3 as an improvement to the existing capabilities approach, which has some drawbacks. The most important benefit of permissions versus capabilities is the fact that permissions are available via the REST API in a more flexible way that capabilities, which are tied to the database.

### 5.8.2 Available permissions

#### 5.8.2.1 Recording

Added in version v4.3. Used to allow administrators and developers to grant voice and video recording permissions to users and domains. Its configuration and scope is explained in the *Recording chapter*

#### 5.8.2.2 Voicemail

Added in version v4.3. Administrators can setup whether an user or domain is able to reach an external voicemail server and see the message waiting indicators. Its configuration and scope is explained in the *External Voicemail chapter*

#### 5.8.2.3 receiveCalls

Added in version v5.12. Administrators can setup if a user or domain users are able to receive calls. Its configuration and scope is explained in the *One to one calls chapter*

### 5.8.3 Configuration

This service needs to be explicitly configurated in the `wac.ini file` by adding a line containing the text `[permissions]`:

```
[permissions]
```

There are no additionals fields to configure in this service.

> **Warning:** Importante note: this service **cannot be deactivated**, even if there are no permissions in use in the system. Make sure that the service is correctly activated in your system, otherwise the calls won't progress.

## 5.9 Managing users

We are going the cover this section the basic actions to handle the users: create, list, modify and delete. All these actions here can be complemented with other actions available on the *REST API*.

### 5.9.1 Adding users

User are created using the Kitter service via its REST API, which will act as a broker between Keycloak (which stores the login information) and wac-core (which stores the business logic).

### 5.9.2 Editing users

As administrator, you need to update users, change its domain, or its personal data (mail, phone, etc). This can be done also by using the Kitter REST API.

### 5.9.3 Delete users

This can be done also by using the Kitter REST API.

This action will simply remove the user entry on the system. In addition, if the SIP mapping service is being used, the SIP mapping associated to that user is removed and the sip mappings pool is marked as "available", as explained in the *SIP mappings* section.

> **Warning:** All other elements linked to this user, like credentials or any other element on the database not detailed above, will not be removed with this command.

### 5.9.4 Adding SIP network access (SIP mapping)

In order to grant a user or *usergroup* to reach the SIP network, it is required to configure properly the SIP trunk and to add a `sip-mapping` to the resource. To do it, add the username of the resource on the SIP network. This will be used to identify the user or usergroup on the SIP network:

```
POST https://wac:8001/sapi/sipMappings/ HTTP/1.1
     Authorization: Bearer ec3fe973ea04739e1...12ec5adc8b323c83c0
     Content-Type: application/json

     {
     "ownerUri": "wac-user:aliceId",
        "credential": {
           "username": "205"
           "password": ""
        }
     }
```

The previous example will create a `sip-mapping` that will identify the user as "205" on the SIP network. Also, any call received on the SIP side with 205 as destination will be routed to the corresponding user (identified by its UUID `wac-user:aliceId`).

### 5.9.5 Auto-provision of users

In some scenarios, the users already exists in another system (typically, an Active Directory server) and they are not pre-provisioned on the Quobis communication platform. In this scenario, the platform can be configured in a way that any user that is authenticated in a trusted third-party system will automaticall have an account in the platform. In other words, user in Quobis wac are automatically created when a valid user, coming from a trusted third-party system, tries to log in for the first time.

> **Warning:** In this case, the Kitter service is not involved and all the activation is done in the wac-core service and Keycloak.

When a user, that has no account in wac-core, tries to log in for the first time, the wac-core service reads it JWT and checks whether this users exists or not in the local database (by reading the username or email value according to the mapping configuration in the auth.userIdentifier parameter in config/auth.toml wac-core file).

If the user does not exist, and the auth.createOIDCUser is set to "true", the following information will be used to create a new user in wac-core by reading this information from the JWT and the auth.toml configuration:

- `username`: read from the JWT data, "username" value
- `domain`: when auth.createOIDCUserForcedDomain is set, this value is used. Otherwise, this value is read from the "domain" property of the JWT
- `mobilePhone`: hardcoded to a empty string
- `email`: read from the JWT data, "email" value
- `capabilities`: taken from auth.OIDCUserCapabilities value in auth.toml
- `role`: set by defaul to "user"
- `displayName`: read from the JWT data, "name" value. Keycloak generates this value by concatenating name & surname.

## 5.10 Agents

### 5.10.1 Introduction

Any user in the system can be considered as an "agent", which means that he/she can be assigned to incoming calls based on a assignment strategy and skills. In addition, agents can have "skills" as introduced below.

**Agent setup**

First of all, it is necessary to register a user in the WAC. Then, it is necessary to identify that user as an agent calling the 'Create agent' API request. In other words, agents are just regular users that have been granted with the "agent" feature. This is made via a POST call to the REST API endpoint https://your-wac-server/sapi/agentSkills/agents.

**Skills**

Skills are abilities or characteristics that may be of interest to define which reduced set of agents is a candidate to receive a call. An example of a skill could be identifying which languages each agent speaks. In this way, when the caller initiates the call, he can indicate in which language he speaks and the candidates to receive that call will only be agents who have the ability to speak the same language.

To manage the skills, there are a group of endpoints that allow us to create or delete a skill, consult the available skills, assign or unassign a skill to an agent. Please check the REST API reference for more information.

### 5.10.2 Agent assignation

The "Agent assigner" attends requests from any source and provides a valid agent as a response following some allocation method. Two different assignation mechanisms are currently supported:

- **system agent assignation**: will assign the first available agent, that is, the first agent connected to the system whose **presence** is 'unknown' or 'available'.
- **external agent assignation**: will delegate the agent assignation to a third-party REST API. A POST request will be made against a previously configured endpoint.

One, and only one, of these mechanisms, could be used at a given time. The resolution method is specified by the administration in the configuration as described below.

The request made from the client to the backend needs to have the following format:

```
{
     channelId: string;
     channelType: string;
     context: object;
     skill?: string;
     token?: string;
}
```

with the following meaning:

- `channelId`: unique identifier for the resource being assigned (roomId, conferenceId, etc);
- `channelType`: one of the following `chat`,``voice``, `c2c`;
- `context`: any arbitrary unknown object intended for usage by the third party assigner that manages the REST API.
- `skill (optional)`: string used to filter which agents can receive the call. Skills must be defined via 'Create AgentSkill' API request
- `token (optional)`: any arbitrary string used for the external agent assignation

There are an existance implementation in our c2c product.

Please note that, in order to be able to call an agent, she needs to be online (presence `online: true` and `activity: available` or `activity: unknown`).

### 5.10.3 Service Configuration

To enable the service the service, this initialization block needs to be added to``wac.ini`` file. .. code-block:: ruby

[agentassigner]

**Note:** It is required to activate the PresenceService as well, since it isn't loaded by default as the core services.

#### 5.10.3.1 External assignation

The mapping service type is configured in `config/agentassigner.toml` file:

```
[agentassigner]
externalAssignerUrl = "url of the external assigner endpoint"
```

The request to the `externalAssignerUrl` will receive a HTTP POST call with the following body:

```
{
     channelId: string;
     channelType: string;
     context: object;
     skill?: string;
     token?: string;
}
```

The response of the `externalAssignerUrl` must have the following body: `{agentId: string}`, where `agentId` will be the identifier of the assigned agent in the WAC. Please note that, for a user to be considered an `agent` with an external allocation method, it is enough that he/his is registered in the WAC.

#### 5.10.3.2 Internal assignation

If the `config/agentassigner.toml` does not exist, is empty or contains an empty `agentassigner` section the internal agent assigner will be used. It is possible to define an agent assignment strategy in the mentioned file as follows:

```
[agentassigner]
strategy? = 'random' | 'linear'
```

- `strategy(optional)` : By default it will be 'linear' in case the parameter has not been defined.
  - Linear: Gets the first agent among those available.
  - Random: Randomly obtains an agent among those available.

### 5.10.4 Logs: known log messages

| Level | Message |
|---|---|
| Silly | `Agents obtained: ${users}` |
| Error | `An error ocurred while resolving agent` |
| Debug | `Message received: ${message}` |
| Debug | `Trying to obtain agent` |
| Debug | `Trying to obtain agents` |
| Debug | `Available agent found: ${agents[0]}` |
| Debug | `Checking if ${agent.id} is available` |
| Debug | `Getting presence for user ${userId}` |
| Info | `Resolved agent ${agent}` |
| Info | `Agent assigner configured with external assigner at: ${externalAssignerUrl}` |
| Info | `Agent assigner will use system assigner` |

#### 5.10.4.1 Troubleshooting & caveats

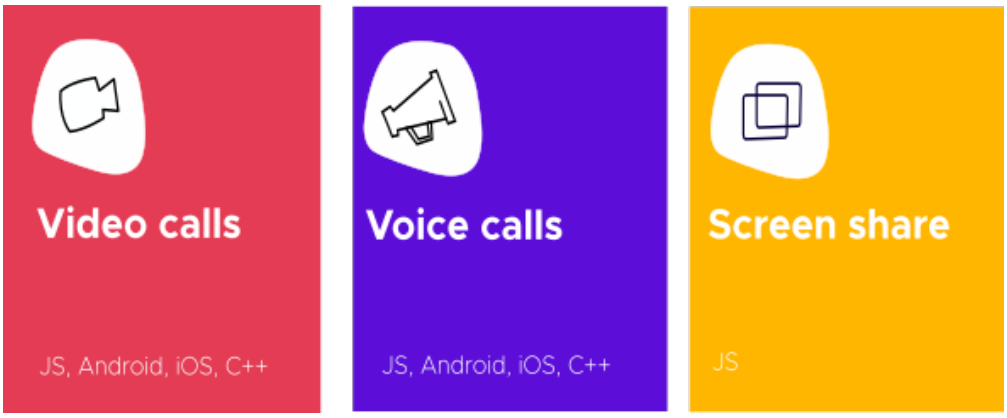
It is not possible to ensure that the response is deterministic. The assignable user offered as return depends on a bulk request, results are not ordered, so *"the first user on the array"* do not always correspond with the same user.

CHAPTER

SIX

# VOICE AND VIDEO CALLS

Quobis WAC allows developers to add real-time voice and video functionalites to Web, Android and iOS applications using WebRTC technology, with extended support to SIP networks and *VoIP and PBX functionalities*. Quobis WAC also supports screen sharing and flexible audio and video recording. It has been designed to be able to adapt to disparate use cases, from traditional one to one calls to multiparty calls, meetings, personal rooms and other complex scenarios.




Quobis WAC makes it simple to developers to create rich and high quality voice and video applications without the need of understanding the complexities of the underlying WebRTC technology. All these functionalities are available via the Quobis SDKs in different platforms, including:

- any modern browser, by using the Javascript HTML5 SDK
- mobile device, by using the native Android and iOS
- embedded devices,by using the C++ SDKs

Quobis wac handles audio and video in a innovate way, using a combination of video forwarding and audio mixing. Developers using Quobis SDK does not need to deal with this complexity as the Quobis wac is in charge of managing this setup automatically, but its worth understanding the basic of how *a call is setup* and the way that *voice and video streams are managed*.

**Note:** If you are already familiar with our product, you can skip this section and jump right into development section by checking the quick start guides and tutorials:

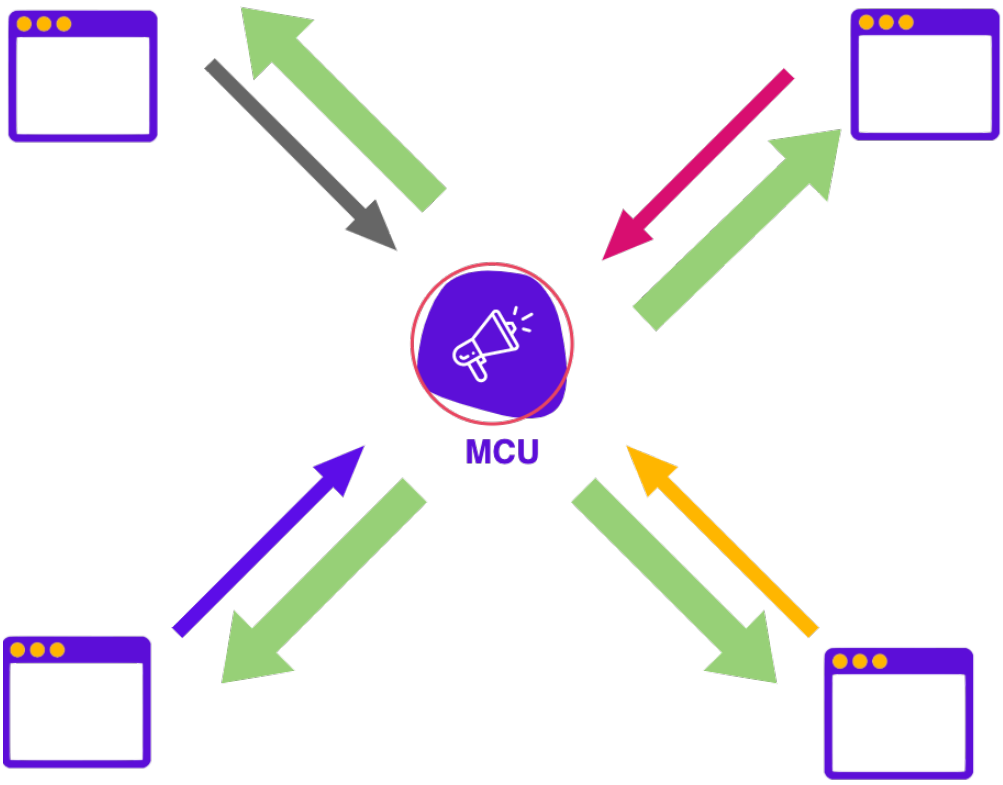
- Javascript tutorials
- iOS tutorial

## 6.1 Media management

Media management is a key discussion point in the overall performance of a real-time application. Quobis wac media server makes a innovative approach to media management which consists of a mixed architecture with an audio MCU (**Multipoint Control Unit**) and a video SFU (**Single Forwarding Unit**) for better and more flexible configuration to address a wide range of use cases.

Audio management

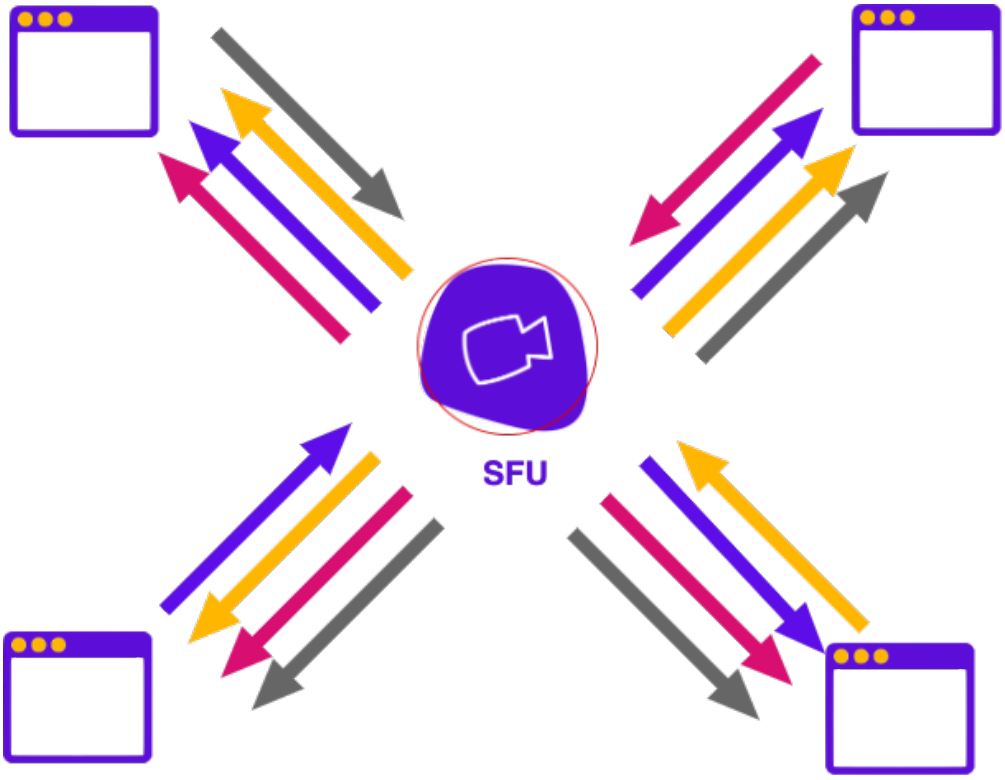
Audio streams are handled by the a MCU (Multipoint Control Unit) which mixes the audio coming from all the participants (also referred as "audiomixer"). Mixing audio is much less expensive in terms of CPU than video and this enables the option to join external SIP calls, typically coming from PSTN to the conferences. Additionally this reduces the bandwidth consumption, since from the web and native clients using the there will be a single bi-directional audio flow which will even take advantage of silence suppression techniques to reduce the effective bitrate.




Video management

Video streams are handled in the video Selective Forwarding Unit (SFU). That means that each participant in a conference will publish a copy of their video stream, and they will subscribe to the videos from the rest of participants. The subscription to receive the video from the rest of participants is done directly against the SFU. This means that, in a conference with N participants, each participant will be publishing one video stream and will be receiving N-1 video streams as shown in the image below. There are a number of advantages of the SFU architecture for video:

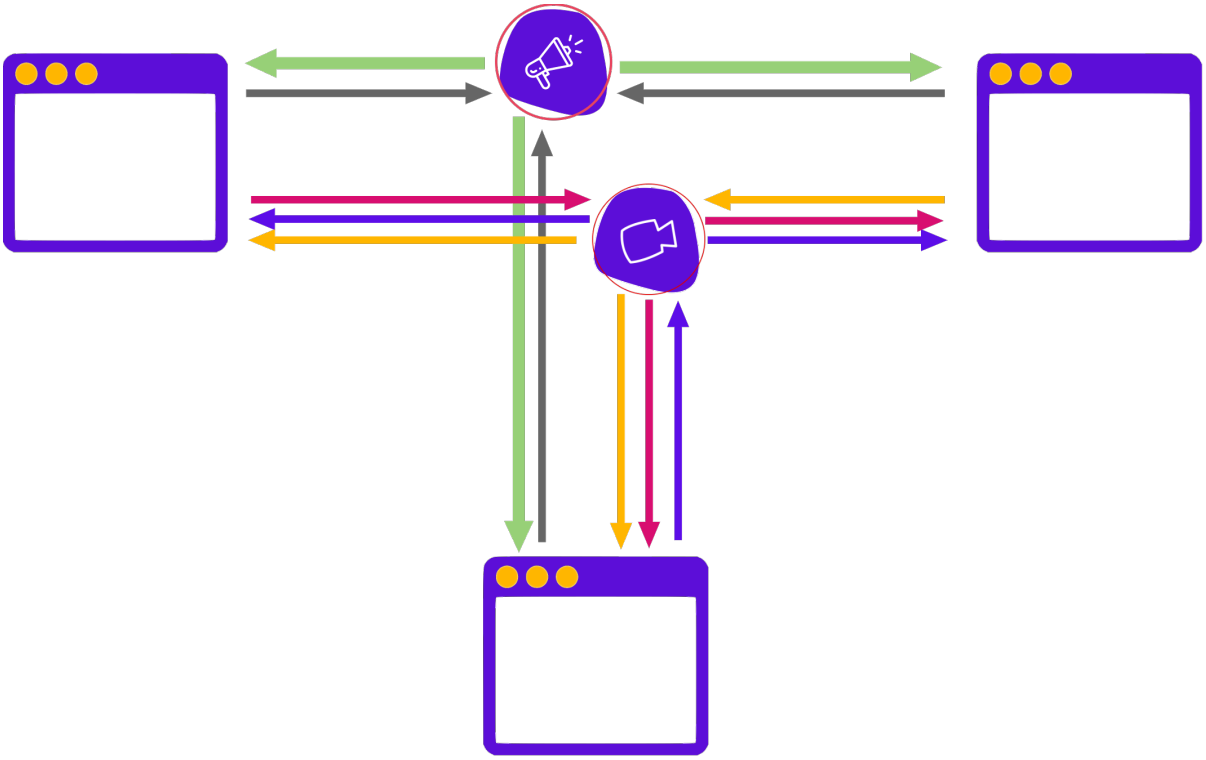
- This architecture is less demanding to the server resources as compared to other video conferencing architectures.
- Since there is only one outgoing video stream, the client does not need a lot of upstream bandwidth
- The endpoint has control of each independent video stream in order to create its own layout
- Reduced CPU load in the server, as video is not manipulated
- Improved scalability and ability to have different video layouts




**Note:** Currently, the SDKs subscribes to all the available video flows. This needs to be taken into account when the number of participants is too high.

How that fits together

As explained above, the media management in Quobis wac media server is twofold: **MCU for audio and SFU for video**. The developers does not need to deal with this complexity as the Quobis wac is in charge of managing this setup automatically, by creating an audio room for each video room. The audio and video flows can be explained by the picture below:



The green arrows represent the audio streams, while the blue, red and orange arrows represent the video streams. The logical entity formed by a joint combination of an audio room (MCU setup) and a video room (SFU setup) is known in our terminology as a "**conference room**" in the Quobis terminology.

A conference room is managed from the *Quobis Signaling Server*. The section *Communication setup* explains how a conference room can be used to implement a wide range of use cases, from basic telephony applications ("A" calls "B") to more complex use cases such as multiparty conferencing, personal rooms, etc...

## 6.2 Media negotiation

This section explains how a participant can negotiate its audio and video streams when joining a *conference room*. The process has two steps: one negotiation for the audio stream and another negotiation for the video stream.

When a user joins a conference room, several media connections are established one for receiving and sending audio, one to send video and then one for each of the participants in the conference publishing video. Each media connection is called a `PeerConnection` in WebRTC terminology. WebRTC uses SDP to negotiate the parameteres of the media connection. SDP requires an offer with the supported media parameters such as codecs, encryption and ICE candidates (which helps to discover the IPs which are going to be used for send and receive media). The media negotiation can be easily debugged with chrome://webrtc-internals when using the Javascript SDK in Google Chrome. All the negotitation happens using the Websocket connection against the Quobis WAC, and it is automatically routed by the internal websocket gateway to the right *video SFU*.

Audio negotiation

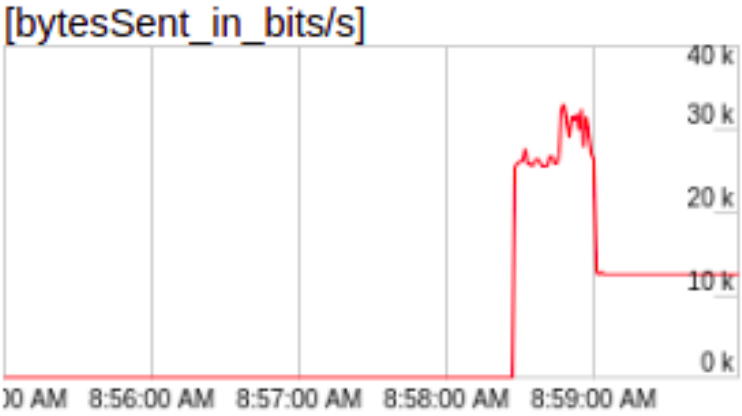
As mentioned above, the audio negotiation only happens onceat the begining of the call and the Peerconnection remains until the user leaves the conference. The PeerConnection for audio is started by the SDK (both web and mobile) which generates a WebRTC compliant SDP offer. This offer is sent to the SFU using the Websocket connection.

```
▼ createOfferOnSuccess

type: offer, sdp: v=0
o=- 2880236297092401525 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE 0
a=extmap-allow-mixed
a=msid-semantic: WMS JkS4xydsjG1fQhzGFcKRJuMkDqQzMxIe1gyc
m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 110 112 113 126
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:nPjO
a=ice-pwd:WhjX2aFJNehz5UpgQOHB3KXE
a=ice-options:trickle
a=fingerprint:sha-256 5F:E0:1C:9B:4A:45:BA:EF:59:C1:BF:71:7A:A1:B2:3E:F2:DC:15:15:B8:72:4F:B2:FF:88:12
a=setup:actpass
a=mid:0
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:5 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:6 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
a=sendrecv
a=msid:JkS4xydsjG1fQhzGFcKRJuMkDqQzMxIe1gyc 46b3be8a-e166-4e67-8ca7-ca509ddfa0c9
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=rtcp-fb:111 transport-cc
a=fmtp:111 minptime=10;useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:110 telephone-event/48000
a=rtpmap:112 telephone-event/32000
a=rtpmap:113 telephone-event/16000
a=rtpmap:126 telephone-event/8000
a=ssrc:845727115 cname:aqUaTAT6BMb/Fr5H
a=ssrc:845727115 msid:JkS4xydsjG1fQhzGFcKRJuMkDqQzMxIe1gyc 46b3be8a-e166-4e67-8ca7-ca509ddfa0c9
a=ssrc:845727115 mslabel:JkS4xydsjG1fQhzGFcKRJuMkDqQzMxIe1gyc
a=ssrc:845727115 label:46b3be8a-e166-4e67-8ca7-ca509ddfa0c9
```

The *audiomixer* is the element which decides which codec to use. The SFU sends the SDP offer to the audiomixer the which processes the offer and generates an SDP reply. By default, Opus codec is replied as the first option but it can be changed in the audiomixer configuration. Once the SDP is answered, the Peerconnection tries to get established.

```
5/12/2021, 8:58:25 A
M
▼ setRemoteDescription

type: answer, sdp: v=0
o=- 11523889 3762317073 IN IP4 54.194.227.143
s=Asterisk
t=0 0
a=group:BUNDLE 0
a=msid-semantic: WMS janus
a=ice-lite
m=audio 9 UDP/TLS/RTP/SAVPF 111 126
c=IN IP4 54.194.227.143
a=sendrecv
a=mid:0
a=rtcp-mux
a=ice-ufrag:TgA8
a=ice-pwd:xSrVFg+M9DelN7QZ/SLdUj
a=ice-options:trickle
a=fingerprint:sha-256 77:87:1C:D8:A8:E9:AB:5F:4F:78:DF:3F:D4:99:E8:51:A9:34:BB:FF:62:06:B2:64:74:1D:2C:EF
a=setup:active
a=rtpmap:111 opus/48000/2
a=rtpmap:126 telephone-event/8000
a=fmtp:126 0-16
a=ptime:20
a=maxptime:20
a=msid:janus janusa0
a=ssrc:2862410266 cname:janus
a=ssrc:2862410266 msid:janus janusa0
a=ssrc:2862410266 mslabel:janus
a=ssrc:2862410266 label:janusa0
a=candidate:1 1 udp 2013266431 54.194.227.143 17874 typ host
a=end-of-candidates
```

The establishment of the media connection will only fail if there is not connectivity which would only happen when the user is behind very restrictive firewalls or HTTP Proxy and the TURN server is not properly configured. Please note that when the audio is muted, the peer connection keeps sending data but a very low rate. This is different for the video, when the video is muted the PeerConnection is closed and a new one is created when the video is unmuted.




Video negotiation

In each conference there will take place several video negotiations:

- One negotiation for the video being published by the user.
- One negotiation for each of the participants in the conference which publish video.

The video negotiation happens between the SDK and the SFU. There are differences in the negotiation when the SDK is publishing the video from the local cameras and when the SDK is subscribing to the video of other participants. When the SDK is pubslihing video, the SDK sends an SDP offer including all the video codecs suported by the SDK (they are not included in the screenshot below).

```
▼ createOfferOnSuccess
type: offer, sdp: v=0
o=- 1938668746225901336 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE 0
a=extmap-allow-mixed
a=msid-semantic: WMS nrQXt3QfbLSOEYV3IRAevPF862Rjfqov1hqk
m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101 102 121 127 120 125 107 108 109 35 36 124 119 123
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:AS9p
a=ice-pwd:gSWvBt26Q/j/Ailz9GfpTDGK
a=ice-options:trickle
a=fingerprint:sha-256 95:33:CC:B1:21:F2:32:5A:76:96:14:04:23:11:A1:1A:3D:7F:63:45:46:12:8D:E5:9B:34:B8:93
a=setup:actpass
a=mid:0
a=extmap:1 urn:ietf:params:rtp-hdrext:toffset
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:3 urn:3gpp:video-orientation
a=extmap:4 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:5 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=extmap:6 http://www.webrtc.org/experiments/rtp-hdrext/video-content-type
a=extmap:7 http://www.webrtc.org/experiments/rtp-hdrext/video-timing
a=extmap:8 http://www.webrtc.org/experiments/rtp-hdrext/color-space
a=extmap:9 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:10 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:11 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
a=sendrecv
a=msid:nrQXt3QfbLSOEYV3IRAevPF862Rjfqov1hqk 26279606-0e1c-4185-8ca3-c89fc392a94f
a=rtcp-mux
a=rtcp-rsize
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 goog-remb
a=rtcp-fb:96 transport-cc
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
a=rtpmap:98 VP9/90000
```

The SFU replies with an SDP answer which defines the video codec to be used. So the codec to be used in the conference need to be configured in the SFU. Currently VP8 and H264 are supported.The codec included in the SDP answer is the going to be used for the video.

```
▼ setRemoteDescription

type: answer, sdp: v=0
o=- 1938668746225901336 2 IN IP4 54.194.227.143
s=VideoRoom 78444861
t=0 0
a=group:BUNDLE 0
a=msid-semantic: WMS janus
a=ice-lite
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 54.194.227.143
a=recvonly
a=mid:0
a=rtcp-mux
a=ice-ufrag:HTPE
a=ice-pwd:I03u2K44FL4eoCy46zNnLq
a=ice-options:trickle
a=fingerprint:sha-256 77:87:1C:D8:A8:E9:AB:5F:4F:78:DF:3F:D4:99:E8:51:A9:34:BB:FF:62:06:B2:64:74:
a=setup:active
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtcp-fb:96 goog-remb
a=rtcp-fb:96 transport-cc
a=extmap:3 urn:3gpp:video-orientation
a=extmap:4 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:5 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=extmap:9 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:10 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:11 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
a=msid:janus janusv0
a=ssrc:689196552 cname:janus
a=ssrc:689196552 msid:janus janusv0
a=ssrc:689196552 mslabel:janus
a=ssrc:689196552 label:janusv0
a=ssrc:2549476806 cname:janus
a=ssrc:2549476806 msid:janus janusv0
a=ssrc:2549476806 mslabel:janus
a=ssrc:2549476806 label:janusv0
a=candidate:1 1 udp 2013266431 54.194.227.143 18122 typ host
a=end-of-candidates
```

In the case of the subscription to remote videos from other participants, the SFU sends an offer (including only the video codec configured in the SFU) and this is answered by the SDK. There will one of these PeerConnection negotitations per participant.

```
▼ setRemoteDescription

type: offer, sdp: v=0
o=- 1620810560968846 1 IN IP4 54.194.227.143
s=VideoRoom 78444861
t=0 0
a=group:BUNDLE video
a=msid-semantic: WMS janus
a=ice-lite
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 54.194.227.143
a=sendonly
a=mid:video
a=rtcp-mux
a=ice-ufrag:kX1n
a=ice-pwd:YiTObDCEvtPbEChqqfNeUf
a=ice-options:trickle
a=fingerprint:sha-256 77:87:1C:D8:A8:E9:AB:5F:4F:78:DF:3F:D4:99:E8:51:A9:34:BB:FF:62:06:B2:64:74:1D
a=setup:actpass
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtcp-fb:96 goog-remb
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:2 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:3 urn:3gpp:video-orientation
a=extmap:5 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
a=ssrc-group:FID 2212356795 3053517469
a=msid:janus janusv0
a=ssrc:2212356795 cname:janus
a=ssrc:2212356795 msid:janus janusv0
a=ssrc:2212356795 mslabel:janus
a=ssrc:2212356795 label:janusv0
a=ssrc:3053517469 cname:janus
a=ssrc:3053517469 msid:janus janusv0
a=ssrc:3053517469 mslabel:janus
a=ssrc:3053517469 label:janusv0
a=candidate:1 1 udp 2013266431 54.194.227.143 18940 typ host
a=end-of-candidates
```

Since the offer includes just one video codec (the one configured in the SFU), the answer must include just that video codec which will be the one used to receive the video flow.

```
▼ setLocalDescription

type: answer, sdp: v=0
o=- 7498841406196658163 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE video
a=msid-semantic: WMS
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:DJaN
a=ice-pwd:N+kpOAje0vPhqildpmkZqAYD
a=ice-options:trickle
a=fingerprint:sha-256 EB:FB:52:C2:A7:17:DC:53:BB:99:6F:B3:47:B0:B7:E7:B0:E3:A9:D5:9B:1A:77:2E
a=setup:active
a=mid:video
a=extmap:3 urn:3gpp:video-orientation
a=extmap:2 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:5 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
a=recvonly
a=rtcp-mux
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 goog-remb
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
```

**Note:** Please note that only **one video codec** can be used in a given conference room. Every participant will use the same video codec.

## 6.3 Supported codecs and media quality

It is key to understand the different codecs used for audio and video communication, so admins can plan & assess the bandwidth needs before going into production.

Audio

There are three supported codecs for audio streams: G711, G722 and Opus. Default codec used is Opus, as it offers a bitrate which goes from 6kbps for narrow-band voice (perceived quality is a bit worse than PCMU) to 510kbps for full-band audio (High Definition Sound). WebRTC supports silence suppression and comfort noise generation by default, so during silence periods only CN packets will be sent and the aggregated BW may be ~50% of the two-way aggregated bandwidth:

- Opus codec: 30kbps ared used for audio (which means ~55kbps on the wire).
- G711A/U: 64kbps which means ~85kbps on the wire.
- G722: Alternative implementation of G711: 64kbps / ~85kbps on the wire.

The codec configuration is made via the audiomixer configuration file located at `audiomixer/configmaps/pjsip-audiomixer-sfu1.conf`. The codec choice is done via the `allow` and `disallow` keywords followed by one or more of the following values (please note that multiple values are also valid when separated by commas, whitespaces are ignored):

| Codec | Value |
|---|---|
| Opus | “opus” (default) |
| G711 a-law | “alaw” |
| G711 u-law | “ulaw” |
| G722 | “g722” |

It is allowed to configure a codec for the WebRTC connections and a different one for the SIP connection(s). Both codecs can be different and adapted to match the requirements of the deployment. Also, each SFUs could use a different codec but this approach is not recommended. These are the sections that need to be modified in the configuration file, an example follows where OPUS is selected for the WebRTC legs while G711-a is selected for the SIP endpoints:

```
# codec selection for the WebRTC legs
[janus-endpoint]
allow = opus

# codec selection for the SIP legs
[kamailio-endpoint]
allow = alaw
```

Video

The following video codecs are supported: VP8, VP9 and H264. Default codec configured in the system is VP8. In order to calculate the expected quality from a video flow we'd need to know at least these three parameters:

- Codec (VP8, H262, VP9, H265, etc.)
- Bitrate (in Kbps)
- Resolution (HD, VGA, QCIF, CIF, etc.)

Bitrate or video bandwidth (without overhead) may vary from a minimum of 100kbps to a maximum of 2000kbps which is the maximum bandwith supported by the browser. Video bandwith consumption will depend on the characteristics of the image, but you can consider the figures below as a reference:

- 1280:720 at 30FPS → 1000 to 2000kbps.
- 640:360 at 30FPS → 500 to 1000kbps.
- 320:180 at 30FPS → 100 to 500kbps.

As a easy-to-remember reference, the total amount of traffic sent (one-way) during one minute of a video call is:

- 1280:720 at 30FPS → between 7,3 and 14,6 Mbytes
- 640:360 at 30FPS → between 3,6Mbytes and 7,3 Mbytes
- 320:180 at 30FPS → between 0,75 and 3,6 Mbytes

In these calculations we are not considering the overhead bandwidth, but in this case it is proportionally less relevant. Screen sharing is sent as video so you can consider similar bandwidth consumptions. The graphic below shows a capture of the WebRTC Chrome internals with a 1000kbps bitrate and 7 frames per second:

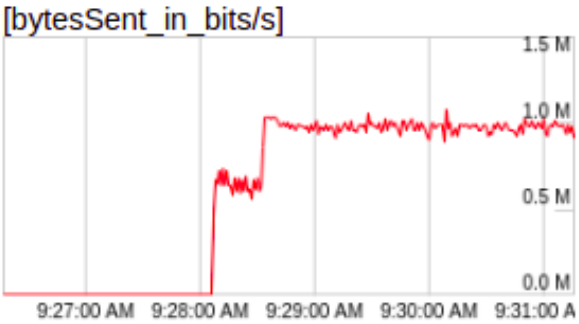
The system has a default configuration value of 300Kbps video bitrate. This value applies to each indivual stream, so take into account that the total used bandwith is not limited by this single parameter alone as there are other participantes in the conference room. The bitrate value can be configured in the parameter `bitrate` available in the dispatcher configuration file `quobis-dispatcher-config.js` (value in kpbs):

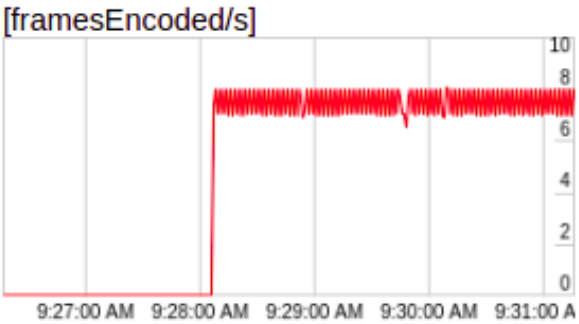
```
config.dispatcher = {
 // Maximum bandwidth that a video stream will have during a call, in kpbs
 bitrate: 300,
};
```

**Note:** Please note that **video transcoding is not supported**, so all the participants in a room must use the same codec, either VP8 or H264.

The list below can be useful when selecting a video resolution:

- HD-720p 1280x720 at 60fps 4096kbps
- HD 960x720 at 60fps 4096kbps
- XGA 1024x768 at 35fps 6144kbps




- WIDE FULL HD 1920x1080 at 30fps 10240kbps
- FULL HD 1440x1080 at 30fps 8192kbps
- HD-720p 1280x720 at 30fps 2048kbps
- HD-720p 1280x720 at 30fps 1472kbps
- HD-720p 1280x720 at 30fps 1024kbps
- HD-720p 1280x720 at 30fps 768kbps
- HD 960x720 at 30fps 2048kbps
- 432P 720x432 at 30fps 1024kbps
- 720x432 at 30fps 768kbps
- 432P 720x432 at 30fps 512kbps
- VGA 640x480 at 30fps 1024kbps
- VGA 640x480 at 15fps 896kbps
- VGA 640x480 at 30fps 512kbps
- VGA 640x480 at 30fps 384kbps
- W288P 512x288 at 30fps 384kbps
- CIF 352x288 at 30fps 768kbps
- CIF 352x288 at 30fps 512kbps
- QCIF 176x144 at 30fps 384kbps
- QCIF 176x144 at 30fps 192kbps
- SQCIF 128x96 at 30fps 128kbps
- SQCIF 128x96 at 30fps 64kbps
- 4CIF 704x576 at 25fps 1536kbps
- 4CIF 704x576 at 25fps 512kbps

- QVGA 320x240 at 20fps 384kbps
- VGA 640x480 at 15fps 896kbps
- CIF 352x288 at 15fps 384kbps
- QCIF 176x144 at 15fps 128kbps
- QCIF 176x144 at 15fps 64kbps
- 4CIF 704x576 at 12fps 1024kbps
- QVGA 320x240 at 10fps 192kbps

Please note that many of these resolutions may not be available in your device. Typically, the WebRTC stack will provide the closer resolution when one non-supported resolution is requested. You can specify a the resolution by setting the right constraints in the SDKs (for example, `LocalMediaHandle` object in the Javascript SDK). Other constraints can be defined using this mechanism following the standard contraints defined by the W3C: MediaTrackConstraints https://developer.mozilla.org/en-US/docs/Web/API/MediaTrackConstraints

**Note:** For more information about WebRTC codes, plese check the following documents:

- https://tools.ietf.org/html/rfc7742 WebRTC Video Processing and Codec Requirements
- https://tools.ietf.org/html/rfc7874 WebRTC Audio Codec and Processing Requirements

## 6.4 Communication setup

This section explains the basics of the mechanisms used by Quobis WAC to setup a real-time media connection between users. Let's analyze first the type of scenarios that are supported.

Call-oriented scenarios

In telephony systems where user A wants to call user B, a direct connection is just setup directly between both parties. This is the traditional concept of a "call", inherited from the old telephony times. The caller is the call initiator. The calle is notified of any incoming call. The callee can reject or answer the call. The caller expects a dialtone while the call is ringing and that the call finishes when either him or the callee hang up. This is also the basics of SIP signaling and VoIP telephony.

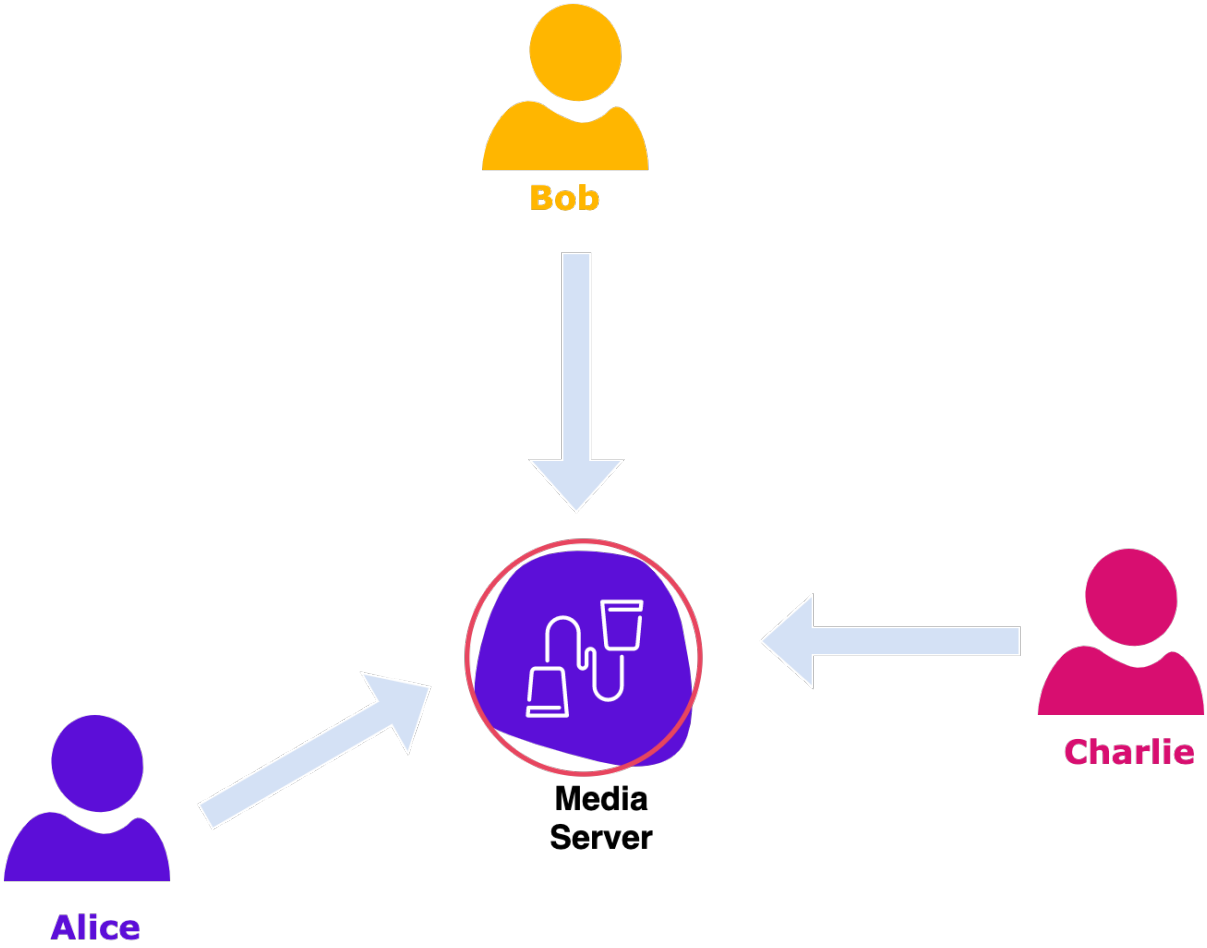
A number of call management functions are available in these systems, such as call transfer, three-way calling, call holding, call forwarding, call parking, call redirection, call pickup, ringing groups, etc. These call management functionalities are provided by the endpoint itself, by the PBX or both. As such, a number of vendors provide a specific SDK and business logic for the "call-oriented scenario".

Besides that, in these scenarios we need to let the users know when there is an incoming call for them, so a method to notify the users need to be put in place as well. This is done in Quobis WAC via a set of *push notification services*.

Conference-oriented scenarios

On the other hand, when the use case requires to connect multiple users together, the common approach is to make use of a common space that the users join (let's call that space "a conference"). From a high

level point of view, this "conference" scenario looks quite different to the "basic call" scenario described above, as there are no "caller" or "callee" roles but a new role of "conference participant". In this scenario, all the participants join the same conference room so they can talk & see each other. For example, in a traditional SIP-based video conferencing systems, every participant dials into a media server which mixes all the audio and video streams together. There is no dial tone and no need for transfers, forwarding, etc... such in the call-oriented scenario.

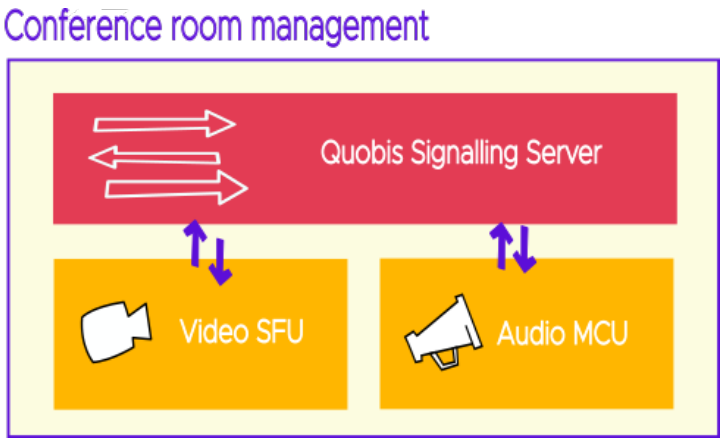



In this scenario, depending on the use case, there might a participant who has specific permissions (usually referred to as "owner") or there might be also users that join the conference room from other networks via a traditional PSTN or SIP call. Participants can also join or leave the conference room at anytime, with or without constrains. In addition, it's up to the users to join the conference room at the agreed time.

Quobis approach

As described above, these two scenarios (call and conference) look like two separate ones in terms of requirements. Because of that, most UC vendors provide separate services, SDKs and APIs for the two scenarios indicated above, each of them having its own business logic and constrains. Hence, the developers cannot mix the both cases together, or event a subset of functionalities. Quobis wac provides an innovative approach that covers **both scenarios with the same infrastructure**. There is no need to know in advance how many users a communication will have, wether it will include or not SIP users or users with mixed audio and video configuration. The architecture to support this approach is based on three main elements:

- a *signaling server* (Quobis Signalling Server, or QSS in short)
- and *media server* which contains:
  - a voice MCU and SIP gateway, to handle the audio streams
  - a video SFU to handle the video streams




Every audio or video connection made with Quobis wac is managed via these three internal components, regardless of the number of users and thus providing the same set of functionalities. These three components are responsible of the setup of "conferences" to communicate users. This **conference-centric approach with support to call-oriented scenarios** is one of the key benefits of the platform and eases the design and implementation of a wide range of use cases thanks to its flexibility.

The beauty of the Quobis SDK lies in this special architecture, where every connection is handled by the same internal components and managed from the *QSS*. In the case of call-oriented scenarios, where there might be only two users, the *QSS* notifies the conference room and the users to "mimic" the telephony events such as ringing, answering, busy-tone, etc... The users perceive that they are in a regular *one to one call* and the fact that they are joining a conference room is transparent to them. This approach provides a strong foundation to developers that gives them freedom to create basic but also quite complex use cases, such as:

- having mixed *SIP and WebRTC users*
- upgrade/downgrade an audio call to a video call
- allow external user to join the conference but also requests users to join a conference
- move a user to another conference, or to a SIP destination
- support to N-way calling (managing more than one conference at the same type for each user)
- automatic calling to N users so they join a conference room when answering
- join users who are offline via desktop and mobile *push-notifications*
- plus all the *telephony features* expected for call-oriented scenarios

There are a number of methods exposed in the Quobis SDKs in order to setup a voice or video communication, all of them built around this same *"conference room"* concept. There is more information about each these methods in the following sections:

- Setup a *basic one-to-one call*
- Setup a conference with multiple participants
- Setup a meeting

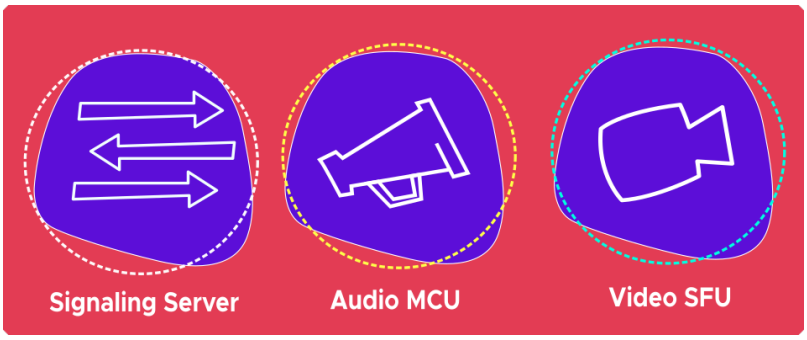
**Note:** Please check section *Media management* for further information and details on how the voice and media streams are actually managed in a conference

#### Summary

The main takeaways of this section are:

- A conference is a logic entity composed of a audio room and a video room, managed from a *signalling service*
- Every connection between users in Quobis WAC is made through a *conference room*, regardless of the number of participants and/or the functionalities needed.
- Quobis WAC provides several services to setup different use cases: calls, meetings and quick meetings.

## 6.5 Conference room




A **conference room** is a logic entity composed of a audio room and a video room, tied together and managed from a *signalling service* and under a number of configuration parameters. The audio room is created in a multi conferece unit (MCU) implemented in a Quobis wac service named *audiomixer*, while the video room lives in a singe forwarding unit (SFU) implemented in Quobis wac in a service called the *video SFU*. Both rooms are "coordinated" by the QSS and other internal services in order to provide a complete audio and video experience to the end-users.

---

**Note:** Summary: a Please pay to attention to the difference between a "audio room" or a "video room" with a "conference room" (which is a combination of a audio room, a video room and a set of parameters).

---

**Every audio or video connection made with Quobis wac is managed via these three internal components**, regardless of the number of users and thus providing the same set of functionalities. This conference-centric approach with support to call-oriented scenarios is one of the key benefits of the platform and eases the design and implementation of a wide range of use cases thanks to its flexibility.

### 6.5.1 Audio MCU

Quobis WAC provides an audio MCU (usually refered to as "audiomixer") based in the Asterisk open source software. The role of the audio MCU is to join together all the audio streams from the participants (both WebRTC and SIP ones) into a single mixed stream that is provided back to all the participants, by making use of the "confbridge" functionality. It is also responsible for the audio recording and interconnection to the *SIP network*.

### 6.5.2 Video SFU

Quobis WAC provides a SFU based in the Janus open source software. The role of the video SFU is to dispatch video stream to the conference participants, and it's also responsible for the video recording. The Quobis WAC -in fact, the *QSS*- takes care of the setup of an audio MCU room and a video SFU room that are tied together in order to setup a conference room. There is a 1-to-1 relationship between the video SFU room and the confbridge instance within the audio MCU. All the SIP calls which can be introduced in the SFU room will also join this same confbridge.

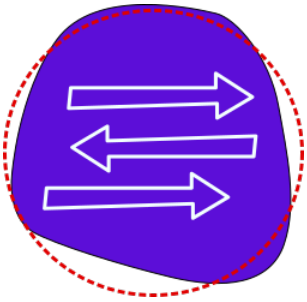
**Note:** The developers using the SDK will not need to deal with these details as the audio/video room management is perfectly wrapped by the SDK code.

### 6.5.3 Properties

A conference room has a set of properties that configure its behaviour and what can be done. Some of these functionalities affect the whole platform while others can be applied to individual conferences.

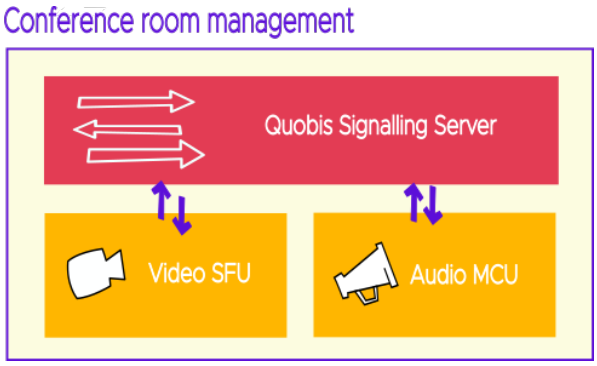
- Access Control List (ACL): this list indicates the user(s) that can access a conference room and that are allowed to publish/subscribe media into it. It is basically a list containing a map of the user ID and a temporary token that it's used internally to create the conference room. There is no limit on the maximum number of users in a ACL and it cannot be changed right from the client application (it's up to the QSS to do that on behalf of the client). On the other hand, there are a number of events that notify the client application about changes in the ACL (see section *"Conference room events"* for a detailed list).
- Room ownwership: a system-wide parameter (`setOwner`) controls wether conference rooms can have an owner or not.If this parameter is set, the parameter `updateRoomWhenOwnerLeaves` needs to be taken into account as well: if set to true, a new owner is chosen from the available users in the conference room when the current owner of the conference room leaves it. Both parameters are set in the *QSS.rooms* service.
- Maximum number of participants: accounts for the maximum number of simultaneous participants into a conference room (regardless of which ones of them are publishing audio streams, video streams or both). Defaults to eight (8) users in case no value for provided in the configuration file (parameter `participantsLimit`). Please note that this is a **system-wide parameter** so it can not be changed for each conference room. When a client application tries to add a participant and this limit has been reached, an error is thrown. This parameter is set in the *QSS.invites* service.
- Allowed audio and video codecs: list of allowed codecs in the conference room. There can be *several codecs*, but only one can be used at the same time. Please note that this is a **system-wide parameter** so it can not be changed for each conference room. If no codecs are provided, default ones are Opus for audio and VP8 for video.
- Bitrate: this parameter, measured in Kbps, defines which is the maximum allowable bitrate per user including every both audio and video streams. The system will prevent the client application to deliver more bandwidth that the one specified in this parameter. This parameter is configurable as explained *here*.
- Recording capabilities: this parameter is configurable for each conference room and can be set to audio only, video only, both or none. More details about how recording works and its implications can be found in the *recording section*.

## 6.6 Call signalling: QSS



The Quobis Signaling Server (QSS) is a service of the Quobis WAC that orchestrates the setup of the voice and video calls. Its main rol is to control the video SFU and the audio MCU in order to create the conference rooms according to the type of service that wants to be implemented. It includes tasks such a create conference room, destroy a conference room, add/remove participants to a conference room ACL, etc... Because of that, the QSS has visibility of the status of each user connection (wether a user has joined a conference room, is put on hold, left the room, etc...)

We use the term signaling, which comes from the VoIP world, to refer to the messages used to let the system know where a specific user is registered and can be found, and all the requests and answers needed to establish calls. For example, when Alice calls Bob, Bob receives a specific type of message called `invite` which will trigger an event in Bob's SDK when the call is received so that this can be answered by him.




QSS features

As the orchestrator of the Quobis WAC calls, the QSS services provides a number of features that developers can leverage to implement a number of different and complex use cases. Some of these features require the interaction of several QSS micro services, while other rely solely on one microservice. These functionalities are exposed in the SDK so the developers can make use of them.

- *Basic conference room management:* these are the basic functionalities that provide low-level control of the conference room. It is suitable when the developers wants to implement a specific use case that is not covered in the standard ones shown below.
- *One to one calls:* this functionality is provided to simulate the telephony use cases, and it's based in the QSS.invite services.
- *Meetings:* this functionality is useful when the call is expected to be scheduled at a certain date and limited to a number of participants.
- *Quick conferences:*: this functionality is useful when we want to create a conference room without any restriction, so it can be reachable by any user at any time.

QSS micro services list

The implementation of the QSS has been made using micro services that communicate between them and with other services via a message broker. Each of these services is responsible for an specific task. In other words, there is no "QSS service" but a handful of "QSS micro services". Being said that, in some

cases we just refer to the QSS as the entity that groups them all for the sake of simplicity. The list of the QSS micro services available in the current version is:

1. **Service QSS.audiomixer-io**: manages the connection with one or more *audiomixer* instances.
2. **Service QSS.audioStatus**: sends mute/unmute requests to remote participants and tracks user mute/unmute status (introduced in v5.6.0)
3. **Service QSS.callTransfer**: handles unattended call transfer between users.
4. **Service QSS.conferenceState**: handles the state of a conference room to decide when it must be destroyed and if a user can be invited.
5. **Service QSS.dtmfAgentProcess**: handles DTMFs generated in a SIP call and, when a combination of tones is reached, calls an user registered in sippo-server and QSS
6. *QSS.invites*: notifies an user that another one wants to invite him to a conference room.
7. **Service QSS.io-websockets**: exposes a Websocket server that the WebRTC clients can connect to.
8. **Service QSS.log-conference**: generates and sends conference objects to sippo-server service
9. **Service QSS.meeting**: allow users to join a meeting if every constrains are met.
10. **Service QSS.peer-jt**: translates and routes messages between different services.
11. *QSS.quickconference*: allow users to join a quick conference.
12. **Service QSS.registry-authenticated**: authenticates and maintains the user sessions.
13. **Service QSS.resolver**: utils for getting more information about a resource, i.e: resolve a user associated with an arbitrary address, get the sip mapping for a user, etc.
14. *QSS.rooms*: manages the creation/update/removal of conference rooms.
15. QSS.trunk: : manages the business logic related with conferences coming/going to/from the SIP network (including the parallel-SIP feature).
16. QSS.watchdog-invites:: removes invitations which are no longer valid.
17. **Service QSS.watchdog-registry**: removes expired register entries and implements a ping-pong mechanism with the clients to know if they are still logged into the QSS.

QSS scalability

Beyond these services, the QSS also interacts with other two internal services, a dispatcher of conferences and a SFU wrapper, in other to properly setup the conference rooms. These services are needed in order to horizontally scale the number of concurrent conferences and to be able to route traffic to several audio MCUs and video SFU. The details of the communication messages interchanged between the QSS and these services are not relevant neither to a system administrator nor to a application developer, so they are not documented here.

### 6.6.1 Conference room services

#### 6.6.1.1 Service name

- service name: SippoAS.qss.rooms-basic
- Service hierarchy: sippoas/qss/lib/services/rooms

#### 6.6.1.2 Functionality

This service manages the creation/update/deletion of conference rooms in the audio MUC and video SFU via the dispatcher service.

#### 6.6.1.3 Service Configuration

The configuration of this service is placed in the config.json:

```
// skipped code
"rooms": {
  "collectionType": "janus",
  "setOwner": true,
  "updateRoomOnwerWhenLeaves": false,
  "janusAddress": "http://dispatcher:8020",
  "removeAfter": 0,
  "participantsLimit": 5,
  "notInACall": true,
  "userIdAsRoomId": true
},
// skipped code
```

Where the meaning of each field is:

- janusAddress: URL of the dispatcher service to be contacted for conference room management.
- removeAfter: wait X milliseconds before removing a peer from the conference room when that peer (a registry object) unregisters or loses connection with qss. Its value as default should be 0 (leave the conference room immediately).
- participantsLimit: max number of participants which can be inside a conference room. When this value is reached, an error is returned when adding a new user to the conference room.
- notInACall: When this flag is false, a user can create as many conference rooms as he wants. If this flag is true, a user will be able to create a new room depending on how the conference state service was configured.
- userIdAsRoomId: When this flag is true, the roomId will be equals to the username of who creates the room.
- collectionType: indicate if it is used the dispatcher or a mongo collection to save data related with the rooms. In order to "janusAddress" makes sense, this key has to be filled with "janus".
- setOwner: If this flag is true, the rooms have an owner and only an owner could perform some actions in the room.
- updateRoomWhenOwnerLeaves: If this flag is true, when the owner of the room leaves it then a new owner is chosen from the available users in the room.

#### 6.6.1.4 Log messages

| Level | Message |
|---|---|
| Debug | *"create"*, peer, record |
| Debug | *"update"*, peer, id, acl, record, dryRun |
| Debug | *"close"*, peer, id, |
| Debug | *"leave"*, peer, id, |
| Debug | *"destroying the room"*, roomId |
| Debug | *"try to know if the user can create a room"*, peer |

#### 6.6.1.5 Troubleshooting & caveats

If the service is not working as expected, a few steps to follow:

- Verify that the message broker is up and running.
- Verify that the service is connected to the message broker.
- Verify that the URL of the sippo-server services API is correct.

### 6.6.2 QSS invites service

#### 6.6.2.1 Service name

- Service name: SippoAS.qss.invites-rooms
- Service hierarchy: sippoas/qss/lib/services/invites

#### 6.6.2.2 Main feature covered

This service is in charge of notifying to a user when other one wants to invite him to a conference room by creating and updating invitations (invite/inviteRequest methods):

- The `invite` method is sent from an user to another peer to invite him to join the user's conference room. This is the method commonly used to start *one to one calls*.
- The `inviteRequest` method is sent to another peer in order to ask for an invite to join receivers room. This method is used for the personal URLs, to include a anonymous user into a existing meeting and to setup incoming calls from PSTN.

The invite services also leverages the push service to send push notifications to mobile devices.

#### 6.6.2.3 Service Configuration

The configuration of this service is placed in the `config.json`:

```
// skipped code
"invites": {
  "basic": {
    "removeAfter": 10000,
    "resolveGroups": false,
    "resolvePush": false,
    "legacyUserGroupResolution": true,
    "wac": {
      "address": "http://wac:8000",
      "username": "admin@quobis",
      "password": "<YOUR_PASSWORD_HERE>"
    }
  },
  "parallelSip": false,
  "onlyOneDeviceWithInvitations": false
}
// skipped code
```

Where the meaning of each field is:

- **resolvePush**: when set to true, invites will try to send invitation by push to the users who are logged in a mobile phone. Will be ignored if **legacyUserGroupResolution** is set to false
- **wac**: sippo-server API URL and credentials to be able to interact with the Sippo AS Push service.

  - **address**: sippo-server URL
  - **username**: administrator user username
  - **password**: administrator user password

- **removeAfter**: wait X milliseconds before removing a candidate from the invite when that candidate (a registry object) unregisters or loses connection with qss. His value as default should be 0.
- **resolveGroups**: when is true, invites service will be capable of retrieving the participants in a group to invite them to a conference. It's essential to perform the 'invite a group' feature. Will be ignored if **legacyUserGroupResolution** is set to false
- **legacyUserGroupResolution**: when is true, invites service will retrieve candidate information as needed from the *sippo-server*, when false it will use the *Reachability* service in the *sippo-server* to get all the candidate information that the *sippo-server* could provide just in one request, significantly improving performance. For backwards compatibility if not specified it is set to true.
- **parallelSip**: This flag enables the parallel sip behavior. When it is activated, an invite is sent to the PSTN net and it come back before inviting to the room to the person who was invited. When this flag is false, invites service works as normally.
- **onlyOneDeviceWithInvitations**: when is true avoids that a user can establish two calls in differents devices. When the user is on two devices (mobile and desktop for example) and has an ringing invite in each device, when the user accepts a invite, the other one will be stopped by this service.

#### 6.6.2.4 Log messages

| Level | Message |
|---|---|
| Debug | `invite peerId(${peer.id}), to(${to}), inviteId(${inviteId}), context(${JSON.stringify(context)})` |
| Debug | `Trying to get candidates for ${to}` |
| Debug | `Trying to resolve group ${to}` |
| Debug | `got candidates ${candidates}` |
| Debug | `Notifying that user ${userId} joins to the room ${roomId}` |
| Debug | `INVITE_COMPLETED will be notified` |
| Debug | `Notifying that invite ${inviteId} for room ${roomId} was accepted by ${whoAccepted}` |
| Debug | `Notifying user ${userId} leaves ${roomId}` |
| Debug | `Cleaning data for room: ${roomIdToClean}` |
| Debug | `accept peerId(${peer.id}), from(${from}), inviteId(${inviteId})` |
| Debug | `decline peerId(${peer.id}), from(${from}), inviteId(${inviteId}), errorCode(${errorCode})` |
| Debug | `"send invitation cancel", invite, "to", candidate` |
| Debug | `"send invitation", invite, "to", candidate` |
| Debug | `"will reject pending invitations of", item, "except", peer` |
| Debug | `"resuming invite", type, peer, message` |
| Debug | `stopInvite with the inviteId ${inviteId} for the callee ${callee}` |
| Debug | `"will update invite candidates with registry", invite, registry` |
| Debug | `"onPeerDisconnected will remove as candidate from", invites` |
| Debug | `No candidates found for resource: ${to}` |
| Debug | `Resource '${to}' is a ${candidatesType} with candidates: ${candidates}` |

#### 6.6.2.5 Troubleshooting & caveats

If the service is not working as expected, a few steps to follow:

- Verify that the message broker is up and running.
- Verify that the service is connected to the message broker.
- Verify that the URL of the sippo-server services API is correct.

### 6.6.3 QSS Quick Conference service

#### 6.6.3.1 Service name

- Service name: SippoAS.qss.quickConference.QConference
- Service hierarchy: sippoas/qss/lib/services/quickConference

#### 6.6.3.2 Functionality

This service is in charge of getting a conference room when a user wants to join one by using a conference URI. This service supplies as well a mechanism to validate if the conference URI used to get a conference room is valid. The conference URI needs to match with the configuration param `uriRegex` defined in the config.json file.

#### 6.6.3.3 Service Configuration

The configuration of this service is placed in the `config.json`:

```
{
 "quickConference": {
                      "uriRegex": "^wac-conf\\d{20}$"
              }
}
```

#### 6.6.3.4 Log messages

| Level | Message |
|---|---|
| Debug | *Peer ${peer.id} wants to join ${conferenceUri}* |

#### 6.6.3.5 Troubleshooting & caveats

If the service is not working as expected, a few steps to follow:

- Verify that the message broker is up and running.
- Verify that the service is connected to the message broker.
- Verify that the URL of the sippo-server services API is correct.

## 6.7 Working with conference rooms

Developers can manage directly the conference rooms to implement the requested use cases via the QSS functionalities. This section explain the main actions that are available to the developers to interact with the conference rooms and the actual messages sent into the wire to ease the troubleshooting.

### 6.7.1 Creating a conference room

This is usually the very first step when we want to setup a communication between two or more users. Any of the users will need to create a conference room by using the `QSS.rooms` service. For example in the Javascript SDK, this is done using the conferenceManager.createConference() method. This is the actual JSON message that is sent from the client application to the QSS. Both the `request` and `requestID` parameters are created automatically by the SDKs, so you don't need to give it a value.

```
 "request":"createRoom",
 "requestID":"zoyhQtp_rbHP_u7TX58GN",
 "payload":{}
```

**Note:** The "requestID" parameter can be used to track and identify the answers to a specific query, as it is the same during each single transaction.

Once the QSS process the message and sets up the conference room, it sends backs a confirmation message to the client application:

```
"id":"60315b4fe951a9c42cae6e0f",
"uuid":"2f11c52047f244f78e1deac1839fbd45d008d20f6eabcf2e17ede0b5d3d520e9",
"owner":"",
"record":false,
"number":"26777033",
"sipUri":"sip:88826777033@audiomixer-sfu1:5080",
"bitrate":1024,
"serverUuid":"RQQVnaPf9ZQtiL2sDDZf5aoZ1gBTUagW6AcIRVG47jyEpl80nKkBMG0be0VmROmf",
"server":{
     "https":"wss://collaborator.quobis.com/sfu1/socket.io",
     "wss":"wss://collaborator.quobis.com/ws-sfu1"
  },
"timestamp":1613847375280,
"participantsLimit":100,
"ice":[{
    "urls":"stun:stun.l.google.com:19302"}],
    "room":"2f11c52047f244f78e1deac1839fbd45d008d20f6eabcf2e17ede0b5d3d520e9",
    "token":"51e4bed51919fe9d4b06a8828c88c19f",
    "participants":["5da6e65772aa8c46876107d6"]
 },
"answer":"createRoom"
}
```

Let's explain each field in detail:

Table 1: CreateRoom message fields

| Field | Description |
|---|---|
| `id` | ID of this QSS invite |
| `uuid` | Unique identifier for the conference room |
| `owner` | User ID of the owner of the conference room |
| `record` | (boolean) Indicates whether the conference is going to be recorded or not |
| `number` | Internal ID used by the wrapper service |
| `sipUri` | Address of the audiomixer confbridge |
| `bitrate` | Maximum allowable bitrate per user |
| `serveruuid` | UUID of the server |
| `server` | Array containing the HTTPS and the WSS address of the wrapper service |
| `timestamp` | UTC, UNIX time in milliseconds of the conference room hour and date |
| `participantsLimit` | Max number of concurrent participants allowed in the conference. |
| `ice` | **Array of ICE servers according to the standard format**<br>• urls<br>• room<br>• token<br>• participants: list of room participants |
| `answer` | Value of the answer (`createRoom` if successful) |

### 6.7.2 Destroying a conference room

Please note that, generally speaking, conference rooms do not have a time to live parameter which means that they remain in the system forever unless they are explicitly destroyed. It's up to the developer to make sure that unused conference rooms are destroyed.

### 6.7.3 Joining a conference room

Once a conference room exists, participants can join it as long as they are in the conference room Access Control List. In case they are not in the ACL, they SDK will thrown an error. For example in the Javascript SDK, this is done using the conference.join() method

### 6.7.4 Leaving a conference room

A participant in a conference room can leave it at anytime. That is achieved by sending a `leaveRoom` message to the QSS along with the conference room ID. That message must include the code error number, which is the value of the SIP error code indicating the cause of leaving ("200" is the default valuee). This is done for example in the Javascript SDK with the method `conference.leave()`.

**Note:** Please note that leaving a conference room does not mean anything for the remaining participant(s), even when there is only one participant left. The conference room exists until it's destroyed.

This is the actual message that is sent from the client application to the QSS:

```
data:
  "{
     "request":"leaveRoom",
     "requestID":"w_RH7mzOLKuebei4ylSO9",
     "payload":{"room":"6d3fa58b1ecfdffd621a3a4e7fbc26d284b6c5e8c8098a0099fa90d95dd86f9a"}
  }"
```

The message sent back from the QSS to the client application is as follows:

```
data:
  "{
      "requestID":"w_RH7mzOLKuebei4ylSO9",
      "cancel":"leaveRoom",
      "error":403,
      "message":"user not included into ACL"
   }"
```

In case that we try to leave a conference room in where we are not part of the ACL, the following message will be received:

```
data:
  "{
      "requestID":"w_RH7mzOLKuebei4ylSO9",
      "cancel":"leaveRoom",
      "error":403,
      "message":"user not included into ACL"
   }"
```

### 6.7.5 Adding an user to a conference room

As explained above, a user who wants to join a conference room and knows its ID can request to join it. In case that the user does not have the conference room ID, it can also be invited to join a conference room by using the QSS.invites service. This functionality is also used when implementing a basic call scenario where a user wants to call another one, as explained in section *"Basic one to one calls"*.

### 6.7.6 Transfer an user to another conference room

Conference room events

There are a number of events available to the developers to notify about changes occurred in the conference room.

Table 2: Conference room events

| remoteStreams | Emitted every time remoteStreams changes |
|---|---|
| outgoingInvitations | Emitted every a outgoingInvitation changes |
| expelled | Emitted when we are expelled from the conference |
| destroyed | Emitted when the conference is destroyed |
| status | Emitted every time the conference status changes |
| remoteParticipants | Emitted every time the participants list changes |
| transferError | Emitted when an error occurs in a transfer |
| participantAdded | Emitted when a new participant joins the conference |
| participantRemoved | Emitted when a participant leaves the conference |
| participantHold | Emitted when a participant starts holding the call |
| participantUnhold | Emitted when a participant stops holding the call |
| localStreamAdded | Emitted on a new local stream |
| localStreamRemoved | Emitted on local stream removed |
| remoteStreamAdded | Emitted on remote stream added |
| remoteStreamRemove | Emitted on remote stream removed |

The QSS offers also other functionalities for common use cases such as *one to one calls*, *meetings* or *quick conferences*.

**Note:** If you are already familiar with our product, you can skip this section and jump right into development section by checking the quick start guides and tutorials:

- Javascript tutorials
- iOS tutorial

## 6.8 One to one calls

**Note:** If you are already familiar with our product, you can skip this section and jump right into development section by checking the quick start guides and tutorials:

- Javascript tutorials
- iOS tutorial

### 6.8.1 Using the "invitation" method to call someone

As explained in section *comunication setup*, the way that Quobis handles the calls always implies the usage of a *conference room*. This also applies for basic one to one calls, where a user wants to communicate with another user. In such a case, Quobis WAC just treats this situation as a conference with "N" participants where N=2, i.e, there are only two users in the same conference room. Let's explain the high-level process that happens when a user (Alice) wants to call another user (Bob):

1. Alice creates a conference room and joins it
2. Alice requests Bob to join its conference room
3. Bob joins that same conference room

One Alice and Bob are in the same conference room they are publishing and subscribing their media so they can start talking and seeing each other. In addition, both Alice and Bob can add more participants to the conference room, transfer another participant, etc. From the point of view of the end-users, the experience is the same as "receiving" a traditional phone call, but what is happening behind the scene

is what we have already described: two users sharing a conference room. This approach gives a lot of flexibility in order to create complex use cases, specially those where we want to add or remove participants "on the fly".

**Note:** There are two important things to highlight here to understand how the "A calls B" process works in Quobis WAC:

- There isn't a direct connection between Alice and Bob, as traditional telephony systems do. Instead, what we have is Alice and Bob talking in the same conference room and a *the QSS* that signals the events that the end-user will expect such as "answering", "ringing", "hanging-up", etc... Alice conference room has a different ID each time a call is placed. In other words, there is no such a thing as "Alice's own conference room" as it changes every time.
- Alice does not "ring" Bob, Alice invites Bob to join a conference room. This is what we call a "QSS invite" message, please no dot confuse it with a "SIP invite" method message as the way it works is different. In other words, when a user receives a "incoming call" what he/she is receiving is an invitation to join a specific conference room.

### 6.8.2 Call flow

The process of setting a basic one to one call between Alice and Bob is summarized in the sequence diagram below. As explained above, the key point to understand here is the use of the **"Invitation Request"** message from the QSS. Please note that a number of other low-level messages and internal processes, such as *audio and media negotiation* have been removed for clarity.

**1.- Alice creates a conference room and joins it**

- A `createRoom` message is sent to the QSS.rooms-basic service. The QSS creates the conference room.
- Alice joins the conference room that has just been created by starting the *audio and media negotiation*.

**2.- Alice requests Bob to join its conference room**

- Alice sends a `invitation` message to the QSS.invites service, telling that she wants Bob to join the room.
- The QSS.invites service resolves Bob's identity and invites Bob to join the conference room and adds Bob to the conference's room ACL

**3.- Bob joins that same conference room**

- Bob notifies the QSS.invites service that he accepts the invitation
- QSS.invites service tells alice that Bob has accepted the invitation and that has joined the conference room
- Bob joins the conference room that has just been created by starting the *audio and media negotiation*.

After these three steps, Alice and Bob are in communication and can talk & see each another. The following ladder diagram shows the actual message interchange.

Fig. 1: Basic one to one call from Alice to Bob

Rejecting a one-to-one call

It can happen that Bob doesn't want to talk to Alice so he rejects the call. In such a situation, Alice will an code error message "603" from the QSS.

```
data: "{
   "answer":"invitation",
   "requestID":"B6yXaxJfBdKlG1K3yI-fX",
   "from":"5da6e65772aa8c46876107d6",
   "error":603
 }"
```

Unanswered calls

It can happen that Bob doest not answer the incoming call. In such a situation, Alice will send this message to the QSS:

```
data: "{
   "type":"request",
   "payload": {
       "endpoint":"close",
       "payload":{"medium":"sipcall"},
       "id":"zMcZeXFM-g9wXKHgZ94ub"
       }
   }"
```

and then receive back this message:

```
data: "{
   "cancel":"invitation",
   "error":408,
   "requestID":"dbUXozlylze1-b8SW1vi5",
   "to":"5da6e65772aa8c46876107d6",
   "from":"5f21f8239797901f01535828"
}"
```

Answering calls with multiple sessions

Our platform supports multi-device and multi-sessions, which means that the user can be logged in more than one device or in several browser tabs at the same time. In such a situation, the behaviour of the system when the user receives and incoming call is as follow:

- If the users **answers** the call in any of the sessions, the other sessions will **stop** ringing
- If the user **declines** the call in any of the sessions, the other sessions will **keep** ringing (and the caller won't be notified)

For example, imagine that we have an user logged via web browser (with an application developed with the Javascript SDK) and is also logged in a mobile app (developed with the iOS SDK, for example). If she reject an incoming call in the mobile app, that incoming call will keep ringing on her PC. If she finally declines the incoming call in her PC, then -and only then- the caller will receive a "User rejected the call" message.

### 6.8.3 Message flow

Let's see the actual message flow in a call between Alice and Bob

Alice creates the conference room so she sends a `invitation` message to the QSS

```
{
 "answer":"invitation",
 "requestID":"BPrV0y-26KQ7hISlENIie",
 "from":"+34986999888"
}
```

Bob receives the invitation message from the QSS with more information:

```
{
   "requestID":"AiYoyeZXKvVa1jDtSPQuX",
   "request":"invitation",
   "to":"5f21f8239797901f01535828",
   "from":"5da6e65772aa8c46876107d6",
   "fromType":"system",
   "mediatypes":{"audio":true,"video":false},
   "context":{"contextInfo":{}},
   "systemFrom":"5da6e65772aa8c46876107d6",
   "payload":{
        "uuid":"1d63845a7b93c9b7ca6b3792ee99928257ac20b18284ded51632958cb3e803df",
        "sipUri":"sip:88814738571@audiomixer-sfu1:5080",
        "owner":"",
        "timestamp":1614031357210,
        "server": {"https":"wss://collaborator.quobis.com/sfu1/socket.io","wss":"wss://
↪collaborator.quobis.com/ws-sfu1"},
        "bitrate":1024,
        "record":false,
        "ice":[{"urls":"stun:stun.l.google.com:19302"}],
        "room":"1d63845a7b93c9b7ca6b3792ee99928257ac20b18284ded51632958cb3e803df",
        "token":"3af66884b2a752f4e73bfa8bd10cbe36",
        "participants":["5da6e65772aa8c46876107d6","5f21f8239797901f01535828"]
   }
 }
```

Bob hangs up the call, so he sends a `leaveRoom` message to the QSS:

```
{
   "request":"leaveRoom",
   "requestID":"1neElgQ3-_hRJ4MS5qcdc",
   "payload":{"room":"8ffbf6016d96ed82019817751afc4b15dd45d36fab46f7d30d40087eb5f332de"}
}
```

Aftewards, Bob will receive the following message, as he's no longer in the ACL:

```
{
   "cancel":"leaveRoom",
   "error":403,
   "message":"user not included into ACL"
}
```

### 6.8.4 Call restrictions

The Quobis communication platform supports blocking the incoming calls using receiveCalls permission assignment.

#### 6.8.4.1 Permission assignment

You can change the receiveCalls permission in order to control the incoming calls, which by default is enable, via request through REST API. A user with this permission disable wont receive incoming calls but still can call other users

#### 6.8.4.2 User

You can disable the receiveCalls to specific users.

To do so, launch the POST request `Add User Permission`, that you can find in the WAC Permissions collection, followed by the user id you want to modify. In the Body you must modify the values of the parameters according to your needs.

Example:

```
curl --location --request POST 'https://web.sippo/sapi/permissions/user/{id}' \
--header 'Authorization: Bearer {bearerToken}' \
--header 'Content-Type: application/json' \
--data-raw '{
    "receiveCalls": {
        "enabled": false
    }
}'
```

Once sent, and if everything has worked correctly, you should receive as a response: 201 Created.

#### 6.8.4.3 Domain

You can disable receiveCalls at a domain level.

To do so use the POST request `Add Domain Permission`, that you can find in the WAC Permissions collection, followed by the domain id you want to modify. In the Body you must modify the values of the parameters according to your needs.

Example:

```
curl --location --request POST 'https://web.sippo/sapi/permissions/domain/{id}' \
--header 'Authorization: Bearer {bearerToken}' \
--header 'Content-Type: application/json' \
--data-raw '{
    "receiveCalls": {
        "enabled": false
    }
}'
```

Once sent, and if everything has worked correctly, you should receive as a response: `201 Created`.

## 6.9 Quick conferences

"Quick conferences" are a special type of conference rooms that have the following characteristics:

- There are no time restrictions
- There is no role of "owner" or administrator
- There is no ACL (Access Control List)

Of course, this use case can also be implemented by directly using the conference rooms methods, but using the QuickConference service is much faster and easier as there is no need of dealing with the conference room ACL, ownership, time constrains, etc... The *QSS.quickconference* is in charge of providing this functionality, which is quite useful when we need to setup a use case that fits the behaviour explained above. A common use case is to provide communication "on the fly" without any restrictions of any type. The Javascript SDK implements the QuickConference service in the `ConferenceManager.getquickconference` method.

Configuration of the QuickConferences is explained *here*.

### 6.9.1 Limitations

Please note that *recording* is not available with QuickConferences. The main reason behind this limitation is the fact that Quickconferences do not have a lifetime, as they are always available, which will mean that recording will last forever. In order to prevent such situation, there recording configuration takes no effect in the QuickConferences. In case that recording is a "must have" feature for your specific use case, please use a standard conference room.

## 6.10 Audio and video recording

The Quobis Communication Platform supports the recording of the audio and video streams. Call recording can be used in a number of situations such as to meet regulatory requirements, verify accuracy, review quality of the customer service, employee training, etc... This page describes the different *configuration* options in order to choose what to record, how to record, who has the right to start a recording and and the characteristics of the recorded files.

### 6.10.1 What can be recorded

Recording of one to one calls and multi-party calls is supported both for video and audio streams. Likewise, if you are developing an application using the SDKs, you can record any conference room that has been created with the *invitation method*. Recording of *meetings public rooms* and is not supported in the current version.

### 6.10.2 Who can record: recording permisssions

Administrators can set up domain-wide and per-user policies in order to decide which users are able to record their outbound calls they make and/or setup mandatory recording by default. In addition, administrators can decide whether incoming calls from PSTN or a SIP trunk are recorder with a system-wide configuration.

#### 6.10.2.1 Permissions for recording of internal calls

The ability to make a recording of a one to one call or multiparty call relies on a set of two *permissions*, named `recordingProfile` and `recordingType`, which are grouped into the recording permission named `RecordingPermissions`. The following table explains the available values of hese two fields:

Table 3: Values of the RecordingPermission set

| Field | Meaning | Values |
|---|---|---|
| `recordingTyp` | Indicates which media streams can be recorded | **The following values are available (only of them)**<br>• `all`: both audio and video streams will be recorded<br>• `video`: only the video stream will be recorded<br>• `audio`: only the audio stream will be recorded<br>• `none`: neither the audio nor the video stream will be recorded |
| `recordingPro` | Indicates the recording policy for this user or domain | **The following values are available (only of them)**<br>• `always`: every call will be recorded by default, regardless of the caller choice<br>• `optional`: let the caller choose whether to record or not<br>• `skip`: calls made from this caller will never be recorded, regardless of the user choice |

Please note that these permissions are related to the **user who starts** the call (the caller) and to the **domain** that this user belongs to. In other words, the decision about whether a call can be recorded or not has nothing to do with the rest of participants involved in the call, it only involves the caller who initiates the call.

The RecordingPermissions are set, updated and retrieved via the following *REST API* endpoints:

- `/permissions`: endpoint to manage **user** permissions. Restricted to the user performing the request
- `/permissions/domain/{id}`: endpoint to manage **domain** permissions. Restricted to the domain administrator

The following code snippet shows and example of a "GET" query answer of the permissions of a user, which are returned in a JSON file format

```
{
  "recording": {
        "recordingType": "all",
        "recordingProfile": "always"
}
```

Please note that the system does not have any permissions created by default.

#### 6.10.2.2 Permissions for recording of SIP calls

The permissions for incoming calls explained above does not apply for the recording of incoming calls. There is a parameter which controls whether incoming SIP calls can be recorded (from PSTN or from a SIP trunk) or not. This parameter (`recordIncomingPSTN`) can be found in the *QSS.trunk service*. If this parameter is set to true, every incoming SIP call will be recorded regardless of the user permissions. If this parameter is set to false, incoming calls from PSTN won't be recorded. Its default value is set to false.

Please note that this is a system-wide parameter and only affects incoming calls, not outgoing calls. It's configuration is explained in the upgrade section.

#### 6.10.2.3 Permissions precedence

In addition to the permissions explained above, there is a generic system-wide parameter in the QSS.rooms services which defines whether the system can record or not in the absence of permissions (in other words, it only applies when there are no permissions configured). This parameter is configured in the config.json file of the QSS configuration as explained *here*

```
... other parameters
record = none, all, video, audio
...
```

with the following available fields:

- `none`: no call will be recorded
- `all`: every call will be recorded
- `video`: video stream will be recorded
- `audio`: audio stream will be recorded

**Internal calls**

The recording permissions apply both to the user and to the domain, being the user permission the most restrictive one. That means that, in case of conflict between the user and domain permissions, the user permissions are the ones that apply. On the other hand, when the user has no recording permissions, the domain permissions apply as a fallback. Beyond that, if there are neither user nor domain permissions, there is a generic configuration that applies to the entire system in the QSS configuration (see section "Configuration" below).

In summary, the precedence for internal calls is as follows: user permissions –> domain permissions —> system configuration. As an example, if a system is provisioned with the following domain permissions, every call made by Alice will be automatically recorded. On the other hand, calls started from other users without any permissions won't be recorded.

- Domain permissions: [recordingType = "none", recordingProfile = "skip"]
- User "Alice" permissions: [recordingType = "all", recordingProfile = "always"]

**External SIP calls**

In this case, there is no precedence as the only valid configuration parameter is `recordIncomingPSTN`. Once a call is set to be recorded, this information is populated in the "createRoom" message and the parameter "record" is set to TRUE.

### 6.10.3 What is recorded

Once a call is set to be recorded, it is recorded from the beginning ("start time") until the end ("end time"). Stopping the recording once it has started it's not possible in the current version. The start time is computed when the callee answer the call. The end time is calculated as follows:

- In **one-to-one calls**, where there is only two participantes, the end time is when any of them hangs up.
- In **multiparty calls**, the end time when one of the last two remaining users leaves the call (which will mean that the system hangs up the remaining user automatically).

The recording includes the actual audio and video streams and metadata such as start time, end time and conference room ID.

### 6.10.4 Output format

Once the call is over, the system will process the audio and video streams and generate the recording files according to the *Service configuration*.. It may take more or less time depending on the configuration options that are explained below. Developers can check that the recordings are ready as a event is generated and sent into the queue as configured in the `processedQueue` parameter.

The system can provide the recording in two output formats:

- Split: each video stream is provided separately
- Merged: all video streams are provided mixed into a single matrix view

Please note that these output formats are mutually exclusive, which means that only one configuration can be active. In addition, this is a system-wide configuration which means that

The default output format, when no one is configured by the administrator, is "merged".

#### 6.10.4.1 Split

This is the most straightforward option, where the video of each participant is provided in a separate video file. These files will have different size in the case of multiparty calls, as not every participant might had spent the same amount of time in the conference. On the audio side, all the audio streams come mixed from the audiomixer so its provided into a single audio file. Additionally, a JSON file is provided to relate all these files together along with the conference metadata. In summary, in a multiparty call with N participants, the system will generate:

- One WAV audio file, containing all the mixed audio streams
- N videos files (webm or mp4), each of them containing the video stream from each participant
- One JSON file with conference metadata

The format of the JSON file is as follows, where we can find the list of video files and the audio file along with its corresponding start and end time.

```
{
 "uuid": "61408347",
 "startTime": "2020-05-25T07:42:28.633Z",
 "stopTime": "2020-05-25T07:42:43.826Z",
 "shouldRecordAudio": true,
 "shouldRecordVideo": true,
 "videos": [
     {
     "startTime": "2020-05-25T07:42:29.126Z",
     "stopTime": "2020-05-25T07:42:43.756Z",
     "filename": "/recording/postprocessed/61408347-c48eb251b24289c951345fa3c57f123e4ee2/
```

(continues on next page)

(continued from previous page)

```
↪rec-20200525-074229-5dde2b2fbd53a5c690.webm",
    "size": "0.123412412341MB",
    "userID": "5dde2b2fbd53a5c690"
    },
    {
    "startTime": "2020-05-25T07:42:36.126Z",
    "stopTime": "2020-05-25T07:42:43.756Z",
    "filename": "/recording/postprocessed/61408347-c48eb251b24289c951345fa3c57f123e4ee2/
↪rec-20200525-074229-074229-5f2av4415c2e44f681.webm",
    "size": "0.323412412341MB",
    "userID": "5f2av4415c2e44f681"
    },
 ],
 "audio": {
    "filename": "/recording/postprocessed/roomId-confbridgeId/rec-20200525-074229-roomId-
↪confbridgeID.wav",
    "startTime": "2020-05-25T07:42:28.633Z",
    "stopTime": "2020-05-25T07:42:43.826Z",
    "size" : "0.23412412341MB"
  }
}
```

The selection of this output format is done by setting `split_recording_output` to *true* in the recording service configuration.

#### 6.10.4.2 Merged

When this output format is selected, the server will generate a single video file containing a matrix that will show all the participants in the screen, adding and removing them according to the time when they have joined or left the conference. This output format requires a post-processing of the raw recording files so it might not be suitable in case that the recordings need to be available right after the call has finished.

The maximum number of participants than this matrix can show is 9 participants (which gives a matrix of 3x3 participants). The number of participants in the matrix has an upper bound that is set by the `mix_limit` parameter. If a conference has more participants than this value, or its value is higher that 9, then the output format will be split instead of merged (even if `split_recording_output` parameter is set to false). Also note that, if `split_recording_output` is set to *true*, the output format will always be splitted.

The following video shows an actual recording of a call with four participants, where you can see how the new participants are added to the matrix:

The selection of this output format is done by setting `split_recording_output` to *false* in the recording service configuration (this is the default configuration).

### 6.10.5 Recording quality

All the *supported audio and video codecs* for making calls are also available for recording. The quality of the recorded files can be configured in order to suit the recording requirements and also taking into account the resulting file size, which can have impact on the storage requirements.

- On the **audio** side, files are stored as WAV file (128 kb/s, 8 bits per sample) quality. As an example, one minute of audio recording generates a 1Mb WAV audio file.
- On the **video** side, an important point that needs to be taken into account is that VP9 provides a better quality when recorded and consumes less disk space, but it processing takes longer that VP8. The recording quality can be configured by choosing three quality levels: low, medium and high.

As a rule of thumb, a one minute video processed with low quality takes 2 MBytes of disk space, while the same video with high quality takes 4Mbytes of disk space.

The following images compare a screenshot of a the same conference call, one recorded with `low quality` (left) and another one recorded with `high quality`, using VP8 codec:

Fig. 2: Screenshot of recording with low quality (left) and high quality (right), using VP8 codec (click to enlarge)

**Note:** Please note that the video size is fixed to 640x480 pixels, regardless of the video quality.

### 6.10.6 Encryption

Recordings can also encrypted for better security and integrity using GPG encryption. In order to use the GPG encryption, the administrator will need to provide a public GPG key. This can be done by using the open source software available at GPG

Decryption is done by using the gpg command with the `--decrypt` option The private key to which the message was encrypted is needed:

```
gpg --output video_unencrypted --decrypt video_encripted.gpg
```

The encrypted recordings are protected even if the servers are compromised and an attacker is able to download the recording files. The encryption takes places immediately after the call is finished. The private key must be protected in a secure device.

### 6.10.7 Service configuration

The configuration of this service is described in the upgrade notes.

### 6.10.8 Available logs

These are the main log messages generated by this service:

- `debug: File encryption is disabled (enabled)`
- `info: FilenameGenerator configured with: {"filenameFormat":"rec-{{uuid}}-{{confbridgeid}}"}`
- `debug: Recording started`
- `debug: Received a process recording event with id: confbridgeID and recording info recording_info.` This log is printed when a conference ends, an event is broadcasted and the postprocessing starts.
- `debug: Processing video: /sippo-recording/raw_files/test/janus-test-userId1-sfuvideo-1615975667821-video.mjr.` For each video in the conference, there will be a log indicating the preprocess of the videos (conversion from .mjr to .webm or .mp4)
- `debug: Trying to encrypt file: /sippo-recording/postprocessed/rec-20200525-074229-test-test.webm`
- `debug: Recording with ID 23423432 processed correctly.`

### 6.10.9 Known limitations

Once a call has started, recording configuration cannot be changed and recording can not be activated or deactivated.

If a user call is recorded, the complete call is recorded: from first user join to last leave is recorded. No matter that a rec-skip user will be involved.

### 6.10.10 Implementation details

This service generates media recordings from the media captured in the *SFU* and *audiomixer*. When a conference room is created with the recording activated, two things happen:

- the audiomixer will save all the audios mixed in a single file (by default, the audio file will be stored in */var/spool/asterisk/monitor* in asterisk machine with name *recording-${confbridgeId}.wav*).
- the SFU will save all the videos flows from all the participants who joined the conference and published video (from a webcam or by sharing the screen).

These files will be persisted in .mjr format (this is a custom format where each file then basically just contains a structured dump of the RTP packets exactly as they arrived). When a conference with recording activated ends, wrapper service will emit an event to the message broker with all the conference information.

Example of such an event:

```
{
  "conferenceId":"conferenceId",
  "recordingInfo":
  {
    "uuid":"conferenceUuid",
    "completed":"2020-05-25T07:42:43.826Z",
    "video":
    {
```

(continues on next page)

(continued from previous page)


```
      "video1":
      {
        "sfuvideo":
        {
          "1590392556275":
          {
            "filename":"/test/test-video-1.mjr",
            "stopTime":"2020-05-25T07:42:43.826Z",
            "startTime":"2020-05-25T07:42:36.277Z"
        }
      }
    },
      "video2":
      {
        "sfuvideo":
        {
          "1590392549123":
          {
            "filename":"/test/test-video-2.mjr",
            "stopTime":"2020-05-25T07:42:43.756Z",
            "startTime":"2020-05-25T07:42:29.126Z"
            }
          }
        }
      },
    "audio":
    {
      "asterisk":
      {
        "sipcall":
        {
        "1590392548633":
        {
          "filename":"audio.wav",
          "stopTime":"2020-05-25T07:42:43.826Z",
          "startTime":"2020-05-25T07:42:28.633Z"
        }
      }
    }
  }
}
```


When the recording services receives this event, it will start processing the recordings. This processs is divided into three steps:

**Preprocessing**

This step is in charge of preprocessing all the .mjr files and converting them to the corresponding media file. To do this, the service will use a binary file offered by janus, called *janus-pp-rec*. Janus-pp-rec actually only dumps the RTP frames it receives to a file in a structured way, so that they can be post-processed later on to extract playable media files. This utility allows you to process those files, in order to get a working media file you can playout with an external player. The tool will generate a .webm if the recording includes VP8 frames, an .opus if the recording includes Opus frames, an .mp4 if the recording includes H.264 frames, and a .wav file if the recording includes G.711 (mu-law or a-law) frames.

After this step, all .mjr files will be converted to .webm files or .mp4 files (in our case, conference could use vp8 or h264 as video codecs).

**Processing Data**

This step is in charge of process all data received in the event emitted by wrapper and generating all the info needed in the postprocessoring to crete the final file. This will generate a list of intervals, where each interval ends when a display layout changes (participant enters or leaves the conference will produce a change in this layout).

**Postprocessing**

This step will received the intervals calculated for the previous step and:

- Create a partial video for each interval received.
- Once all partial videos are created, the postprocessor concatenates them in a single video file.
- Then mix concatenated video with the audio generating the final file.
- Move final file to destination folder.
- If clean_temporary_files is activated, all temporary files related with processing will be deleted.
- An event will be emmited to rabbitmq indicating the processed conference.
- Ack to rabbitmq to remove this task from it.

**Things to keep in mind**

Some things to keep in mind:

- If the recording services fails in any of the previous steps, it will emit an event to rabbitmq (in the *failure_queue*) indicating the error and the conference that fails. In this case, although the clean_temporary_files is active, the temporary files will not be deleted, in order to do not lose these files without having a processed final file.
- If conference has no audio (or some problem ocurred saving the audio by the asterisk), a final file will be generated with all the concatenated videos but without audio.
- The processed file will have this format: *recording-${confbridgeId}-${roomUuid}.webm*
- For the recording to work correctly, it will need to have access to the media files persisted by the audiomixer and by the SFU (this can be done created shared volumes between containers, using a NFS network...).

### 6.10.11 System configuration

The following parameters in the *QSS* `config.json` configuration file defines what to do regarding recording when there are no permissions defined in the system:

- Default value for "record" parameter `none`.
- Default value for "recordIncomingPSTN" parameter is `false`

```
// skipped code

"rooms": {
    "record": all,video,audio,none
  },

“trunk”:{
    “recordIncomingPSTN”: true
 }
```

The following parameter is needed in the *audiomixer* `confbridge.conf` configuration files:

```
// skipped code
  record_file_timestamp = no
// skipped code
```

The following parameter is needed in the *audiomixer* `modules.conf` configuration files:

```
// skipped code
  load = app_mixmonitor.so
// skipped code
```

The following parameters are needed in the recording service configuration file `recording.toml` located at `sippo-recording/config/`:

```
{
  "recordings_path": "/sippo-recording/raw_files/",
  "audio_path": "/sippo-recording/raw_audio/",
  "destination_path": "/sippo-recording/postprocessed/",
  "processing_queue": "processing_queue",
  "failure_queue": "failure_queue",
  "processed_queue": "processed_queue",
  "webm_codec": "vp8",
  "quality": "low",
  "clean_temporary_files": false,
  "fileformat": "rec-{{year}}{{month}}{{day}}-{{hours}}{{minutes}}{{seconds}}-{{uuid}}-{
↪{confbridgeid}}",
  "should_encrypt"  : true,
  "encryption_key"  : "/sippo-recording/encryption/key.asc",
  "encryption_mail" : "mail@domain.com",
  "split_recording_output" : true,
  "mix_limit" : 9
}
```

The default.toml config file needs to have the following fields:

```
[log]
level = "debug"
relativePaths = true

[rabbitmq]
address = "amqp://rabbitmq"
```

The meaning and default values of each field is explained in the table below:

Table 4: Configuration options for the recording service `recording.toml`

| Field | Comment | Default value |
|---|---|---|
| `fileformat` | Available fields that will compose the filename of the recordings, must be placed between double brackets: year, month, day, hours, minutes, seconds, uuid and confbridgeid | rec-{{year}}{{month}}{{day}}-{{hours}}{{minutes}}{{seconds}}-{{uuid}}-{{confbridgeid}} |
| `logs_path` | Path where the logs of the post-process will be saved. | /sippo-recording/logs/ |
| `recordings_path` | Path where the raw video recordings will be stored. | /sippo-recording/raw_files/ |
| `audio_path` | Path where the raw audio recordings will be stored. | /sippo-recording/raw_audio/ |
| `destination_path` | Path where the post-processed file will be saved | /sippo-recording/postprocessed/ |
| `webm_codec` | Codec that will be used when processing a video that has been recorded in VP8 or VP9. Choices: vp8 or vp9. Please note that the final codec is always WEBM | vp8 |
| `quality` | Quality of the processed videos. Choices: low, medium or high | low |
| `processing_queue` | Name of the queue used for receive processing events | processing_queue |
| `processed_queue` | Queue where correctly processed messages will be placed | processed_queue |
| `failure_queue` | Name of the queue used for failures events | failure_queue |
| `clean_temporary_f` | Boolean field. If it is true, when a recording is processed correctly, all clean_temporary_files related with this conference will be deleted. | true |
| `should_encrypt` | Boolean field. If it is true, encryption is applied to the final processed recording files. | false |
| `encryption_key` | Path of the PGP encryption key. | /sippo-recording/encryption/key.asc”, |
| `encryption_mail` | Email address used for the encryption process. | |

CHAPTER

SEVEN

# TELEPHONY AND SIP INTEGRATION

## 7.1 PBX features

Although the Quobis Communication Platfrom does not intend to replace a PBX service, it provides a number of basic PBX-like functionalities that developers can leverage when implementing their *applications*. The most relevant functionalities are:

### 7.1.1 Direct call transfer

Sometimes also referred as "blind" transfer, where a user can transfer a call to another user without actually initiating a call to the end destination.

### 7.1.2 Attended call transfer

Sometimes also referred as "consultative transfer", an attended transfer is a transfer where before actually transferring to the destination, the call is put on hold and another call is initiated to confirm whether the end destination actually wants to take the call or not. These two calls are then merged together.

### 7.1.3 Call waiting notification

Alerts the user of an incoming call while participating in another call, and allows the user to switch back and forth.

### 7.1.4 Call holding

A user can put a call on-hold, recover it or even direct-transfer it to another party. It is also possible to have more than one call holded at the same time. This feature is available as of today in the Javascript SDK only.

### 7.1.5 N-way calling

A user can add another one or more users to a current call, which will result in a multiparty conference.

### 7.1.6 Ringing groups

These are groups of users that receive an incoming call at the same time, which will be connected to the first user that answers. This requires the setup of a *user group*.

## 7.2 SIP integration

This chapter is dedicated to the SIP integration capabilities of the Quobis communication platform. The Quobis communication platform provides integration for voice and SIP based on a dedicated roles on the Media Server focused on the media handler. So WebRTC and SIP are integrated as a single entity. There are several ways to integrate the platform with an existing SIP network:

- **SIP trunk integration**: standard integration based on inbound/outbound calls, providing frictionless integration between WebRTC applications and an external SIP proxy.
- **Parallel SIP**: a specific integration designed to lead the control of the dialplan and routing actions to an external SIP proxy.

### 7.2.1 SIP trunk integration

The SIP integration is done by the SIP proxy service. The platform users can be exposed to the SIP network using the *SIP mappings*. Standard integration based on inbound/outbound calls, providing frictionless integration between WebRTC applications and an external SIP proxy. The platform users will be identified on the SIP network with a SIP identity covered by the `sipmappings` service. During installation of the platform a SIP trunk is established manually on the SIP-proxy, to do that, you should configure your environment properly using the system configuration variables.

In summary, to enable the SIP integration you need to:

- Enable the *sipmappings* service
- Provision sip-mappings to your users as explained in the *user edition* section
- Configure the SIP-proxy and Audiomixer IPs and ports as explained in the *installation section*
- Check *Supported codecs and media quality* on the SIP/media section.

#### 7.2.1.1 Routing of incoming calls with a SIP trunk setup

When a incoming call is received in the Quobis WAC via the SIP trunk, the QSS service will check the "To" field in the SIP invite message and will try to resolve its destination internally in order to know which user should be called. The QSS service will lookup for a SIP-mapping in the wac-core database that matches the "To" field:

- If no match is found, the call will be then rejected with a "404 NOT FOUND" SIP message to the remote SI party and no user will be called.
- If there is a match, the QSS will route the call to the owner of this SIP-mapping. This owner can be an user (identified by a wac-user:uuid string) or a ringing-group (identified by a wac-group:uuid string).

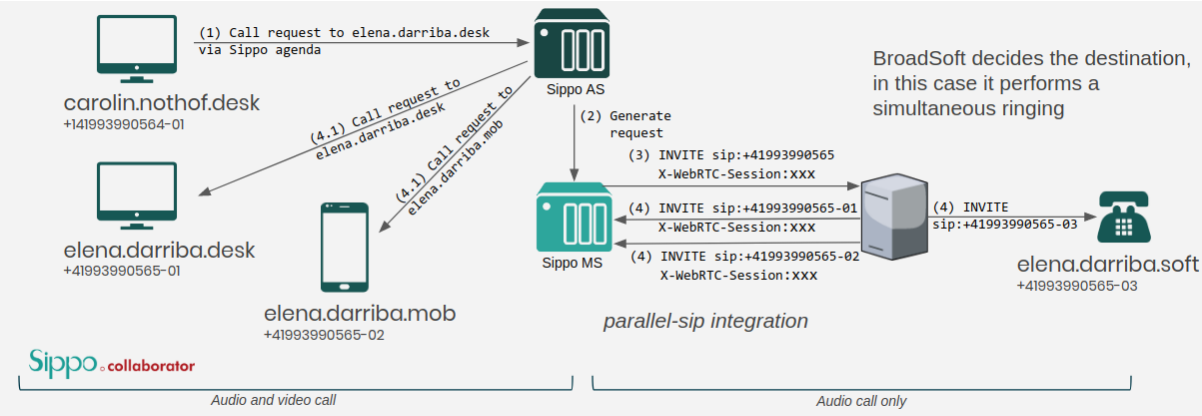
#### 7.2.1.2 Routing of outgoing calls with a SIP trunk setup

When a Quobis WAC user dials into a destination that does not match with another user or userGroup (for example, dialing a phone number) the QSS will check if this call can be routed via the external SIP trunk. In order to do so, the QSS.trunk service will apply a regex expression as set in the "trunk.match" configuration as explained here.

- If the regex does not match, that means that the call is not allowwed and the QSS will reject the call with a "xxx" error code.
- If the outgoing regex matches, the call is routed via the external SIP Trunk, setting TBC as the "From" field in the SIP message. In addition, if the user has a SIP mapping, this SIP mapping information will be set instead.

### 7.2.2 Parallel SIP integration

Parallel-sip is a Quobis Media Server feature that allows to reflect all Quobis application traffic on the customer SIP network. When this feature is enabled, the calls between WebRTC users are sent via SIP to VoIP network and, if the external network decides to forward the call to another WebRTC endpoint, Quobis platform will receive the call and locate the WebRTC user. Mainly it delegates all routing decisions (usually taken internally) to an external VoIP network. The benefits of this configuration are full SIP dialplan control, billing and control on a single point, signaling and traffic penalization and RTP and audio inspection. The following example shows a multiple-ringing scenario where some endpoints are on the provider network (VoLTE for example) and others on the Quobis platform (desktop or mobile apps).




This SIP integration method starts with the standard SIP trunk integration, but it requires to enable the parallel SIP integration parameter.

- Enable the *sipmappings* service
- Provision sip-mappings to your users as explained in the *user edition* section
- Configure the SIP-proxy and Audiomixer IPs and ports as explained in the *installation section*
- Check *Supported codecs and media quality* on the SIP/media section.
- Enable `activateParallelSip` variable as explained in the installation section

This will enable the SIP signaling customization detailed on the SIP integration done by the *QSS service*. The outgoing INVITE request has a specific custom SIP header "X-WebRTC-Session" that contains an uuid, if it is returned in the incoming INVITE from SIP leg then Quobis can match the call from a WebRTC client and will add video using the Quobis MS. Quobis MS will send the audio to BroadSoft, so it is possible to manage a conference room in the SIP side or in the Quobis MS side.

## 7.3 SIP mappings

### 7.3.1 Introduction

The SIP mapping feature provides a way for defining, obtaining and updating SIP identities (`sip-mapping`) associated to a system resource. Supported resources are users and user groups. This feature plays a key role on the SIP integration. It can identify the originator endpoints on the SIP network when they come from the system, and can identify resources on the system based on a SIP R-URI. With this feature, currently individual users and groups could be identified on the SIP network, but it opens the ability to establish multiple politics for SIP assignation.

This feature is implemented in two wac-core services: "SIP mappings" (`sipmappings`) and "SIP identity pool (`sipidentitypool`).

### 7.3.2 Concept

A SIP mapping presents the equivalence between a SIP identity (user@domain) and a system resource (typically an user). By having this equivalence, we can route incoming SIP INVITE methods to a specific resource and replace the system identity when a system invite is being routed to the SIP network.

- **System resource**: Currently the service supports `user` and `usersgroup` as assignable resources, where a `user` is a single platform user and a *user group* is a reachable element that involves multiple participants as a ringing group.
- **SIP mapping owner**: System resource that owns a SIP mapping.
- **SIP identity**: a pair of a SIP username and its associated password. Please do not confuse this SIP username (eg: alice@corporate-pbx.com) with the system username (eg: wac-user:alice_ID)

The following image summarizes the concept of a SIP mapping as a group of a user/usergroup and a SIP identity:

### 7.3.3 SIP mapping assignment

#### 7.3.3.1 Manual

Admins can assign/edit/remove a mapping to a resource at any time by using the REST API endpoint:

- Create a SIP mapping identifying the owner and the SIP identity to use. `POST /sipMappings`
- Update a SIP mapping. Changing both ownership and/or SIP identity. `PUT /sipMappings`
- Obtain the SIP mapping for a given owner. `GET /sipMappings`
- Remove a SIP mapping entry. `DELETE /sipMappings`

When a new `sip-mapping` is created, the service validates if that sip-mapping exists before. An error will be thrown if this sip-mapping already exists. In addition, one and only one `sip-mapping` can be associated to a given user or userGroup.

When a user is deleted from the system, its associated sip-mapping is also deleted (if any).

There is a special case here regarding the anonymous users. This service also maps **anonymous users** with "anonymous SIP identities", that are dynamically generated under request. A SIP mapping for an anonymous user will follow this format:

```
dynamicMappingPrefix + userId + @ + dynamicMappingDomain
```

Where `dynamicMappingPrefix` and `dynamicMappingDomain` are both configurable parameters as explained below in the configuration section.

#### 7.3.3.2 Automatic (pool-based)

Note: this feature has been deprecated since version 5.8.

Since version v4.4, Quobis wac can automatically assign SIP mappings to system resources from a "pool" by using the "SIP identity pool" service. The assignment is done from selecting SIP identities from a pool, so the admin does not need to assign SIP mappings manually via the REST-API. This pool can be populated via the the REST API by using the /sipIdentitiesPool endpoint. There is one pool per domain and each pool can have as many entries as needed. The available fields for each entry are:

Table 1: Parameters of a SIP identities pool entry

| Parameter | Type | Value |
|---|---|---|
| "username" | string | Username of the SIP identity. Do NOT confuse with the wac user name. |
| "password" | string | Password of the SIP identity for the "username" user |
| "targetDomain" | string | This SIP identity can only be assigned to a user of this "targetDomain" domain |
| "available" | boolean | This field indicates if this entry is in user or not. |

This is how the automatic assignment works:

- When a user logs into the system:
  - A SIP mapping from the pool is given to him/her if he/she does not have a associated SIP mapping. In case that the user already has a SIP mapping, nothing happens and the pool remains untoched.
  - Once a SIP mapping is associated to a given user, it's not removed from the pool but marked as "not available" (the "available" field is set to "false").
  - The assignment process works with a FIFO scheme: the first available entry in the pool is the one selected for assignment.
  - In case that there are no available SIP mappings in the pool, an error message is thrown and no assignment is done.
- When a user is removed from the system, two things happen:
  - If the user has a SIP mapping, it is removed from the system as well by the SIP mapping services (this happens even when the service "SIP identity pool" is not running)
  - It that SIP mapping exists in the pool, it's marked as "available" (but it's not removed, so it can be reused later on)

If the "Sip Identities Pool" service is not activated, no SIP mapping is assigned to the user after a successful login. In any case, the administrator can decide to manually assign a SIP mapping via the REST API at any time.

Let's explain the process with an example. Let's suppose that we have a system with two user domains (domain_1 and domain_2) and that have the SIP identity pool populated like this:

Table 2: Initial state of the SIP identities pool

| Username | Password | Domain | Available |
|---|---|---|---|
| 0034986111222@sip-pbx.com | changeme!please | domain_1 | false |
| 0034986111333@sip-pbx.com | changeme!now | domain_2 | false |
| 0034986111444@sip-pbx.com | changeme!hurryup | domain_2 | true |
| 0034986111555@sip-pbx.com | changeme!soon | domain_2 | true |

In this situation, imagine that a new user "alice@domain_2.com" logs into the system. As Alice has no associated SIP mapping, the SIP identity pool service gives her the first available SIP identity from the pool (the one with username "0034986111444@sip-pbx.com") and the status of the SIP identities pool changes to:

Table 3: State of the SIP identities pool after Alice's login

| Username | Password | Domain | Available |
|---|---|---|---|
| 0034986111222@sip-pbx.com | changeme!please | domain_1 | false |
| 0034986111333@sip-pbx.com | changeme!now | domain_2 | false |
| 0034986111444@sip-pbx.com | changeme!hurryup | domain_2 | **false** <——— |
| 0034986111555@sip-pbx.com | changeme!soon | domain_2 | true |

If we perform a GET query to the /sipMappings endpoint passing alice@domain_2.com as the "ownerUri" parameter, we'll get this answer:

```
{
  "username": "0034986111444@sip-pbx.com",
  "password": "changeme!hurryup"
}
```

Some considerations to take into account when using this "SIP identity pool" service are:

- The pool does not contain any entry after the initial setup, so it needs to be populated via the REST API.
- Please note that the service does not check for duplicate entries. That means that it's up to the applications to check if a entry already exists before creating a new one in order to avoid duplicates and other conflicts.
- Please note that the service does not perform a sanity check on the username or password fields. It's up to the administrator to make sure that the fields are correctly created (for example, the "username" parameters must be a valid SIP URI username like "alice@corporate-pbx.com").
- Please note that the service does not check that a SIP mapping is already in use when assigning it to a user. It's up to the administrator to make sure that SIP mappings are not duplicated.
- Please note that, in case an entry in the pool is updated or deleted via the REST API, no actions will be made if these credentials are assigned to an user.

### 7.3.4 Service configuration

#### 7.3.4.1 SIP-mappings service

In order to enable the SIP mappings service, you have to add the configuration block `[sipmappings]` to the `config/wac.ini` file. This service accepts two parameters, both used to configure the format of the dynamical mappings generated to the anonymous users, as explained in the service description.

- dynamicMappingPrefix: prefix used for anonymous SIP identities.
- dynamicMappingDomain: domain used for anonymous SIP identities.

```
[sipmappings]
dynamicMappingPrefix = <PREFIX>
dynamicMappingDomain = <DOMAIN>
```

An an example, the following configuration option:

```
[sipmappings]
dynamicMappingPrefix = anonymous-
dynamicMappingDomain = anonymous-domain
```

will configure the system to assign to an anonymous user with the id: `abc`, the following SIP identity: `anonymous-abc@anonymous-domain`

#### 7.3.4.2 SIP-pools service

In order to activate the automatic assignment service using the pool, the configuration block `[sipidentitiespool]` needs to be included into the `config/wac.ini` file (no extra configuration parameters are needed):

```
[sipidentitiespool]
```

**Internal services interaction**

Other internal services can reach *SipMappings* using the *Message Broker*. For instance, when the Trunk service (`sippo-server.trunk`) requires to identify a system resource, it sends a message to *SipMappings* service in order to obtain the given resource.

**Note:** If `sippoAS.sippo-server.UsersGroup` is not enabled this service will reject any request that try to obtain the `sip-mapping` of a `usersgroup`.

## 7.4 External voicemail integration

The Quobis communication platform supports the integration with external SIP-based voice mail servers. It has the ability to connect to a external voicemail system and show that messages are waiting and the number of them. This functionality is commonly referrered in telephone networks as `Message Waiting Indication` and is usually implemented in IP phones (lighting a light or other indicator on the phone, displaying icons or text, or some combination of them).

For this functionality to take place, it's required that the voicemail system supports the SIP message protocol and specifically the SIP SUBSCRIBE methods as described in the RFC 3842 (https://tools.ietf.org/html/rfc3842). In summary, the Quobis wac will act a SIP User Agent and send SIP SUBSCRIBE message to the PBX or SIP Server, which will answer in turn with SIP NOTIFY messages describing the number of messages pending as show in the SIP trace below:

```
NOTIFY sip:8163689132@63.35.190.183:5064 SIP/2.0
Via: SIP/2.0/UDP 192.104.111.37:5062;branch=z9hG4bK2sansay632037837rdb15824
To: <sip:8163689132@192.104.111.37:5062>;tag=f3b956f774c700489ca8ca889ee831ea-4cf3
From: <sip:8163689132@192.104.111.37:5062>;tag=sansay632037837rdb15824
Call-ID: 4021fa0c43b6f126-8@172.18.0.11
CSeq: 2 NOTIFY
Contact: <sip:8163689132@192.104.111.37:5062>
User-Agent: IVR TalkingSIP/3.15.1.0
Subscription-State: active;expires=1800
Event: message-summary
Max-Forwards: 69
Content-Type: application/simple-message-summary
Content-Length: 87

Messages-Waiting: no
Message-Account: sip:8163689132@10.7.100.16
voice-message: 0/0
```

### 7.4.1 Enabling services

The first thing to do is to enable and configure the services on which this functionality depends, in case they are not already configured by default.

#### 7.4.1.1 Message Waiting Indicator service (MWI)

The activation of the voicemail functionality relies on a this new service named `messagewaitingindicator`. This service requires two configuration parameters: `baseUrl` (the URL where the register and unregister HTTP requests should be sent) and `kamailioRabbitTopic` (RabbitMQ topic where the voiceMail events will be published). The following lines must be added to the `wac.ini` config file in order to run this service:

```
[messagewaitingindicator]
baseUrl = "http://kamailio:8080"
kamailioRabbitTopic = "kamailio"
```

#### 7.4.1.2 Permissions service

In addition to the MWI service, the `permissions` service also must be running to enable or disable voicemail on domain and/or users. To do so, the following line must be added to `wac.ini` config file:

```
[permissions]
```

### 7.4.2 Extension settings

Next, you need to configure the phone extension that will be associated to the external voicemail. To do this, just modify the QSS configuration file `config.json` in the invites section, as indicated in the following lines:

```
{
    "service": {
        //other services
        "invites": {
            //other param configs
            "voiceMailExtension": "XXX"
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

Where XXX represents the phone extension that you want to assign to the voicemail. This extension will be reachable as "xxx@domain", and calls made to that extension will be routed to the voicemail.

**Note:** Please take into account that the voicemail extension number is a system-wide parameter (per-domain configuration is not available in the current version)

### 7.4.3 Permission assignment

The next step is to assign the necessary permissions to start using voicemail, which by default is disabled. This configuration is done through requests to the REST API.

#### 7.4.3.1 User

You can activate the voicemail to specific users.

To do so, launch the POST request `Add User Permission`, that you can find in the WAC Permissions collection, followed by the user id you want to modify. In the Body you must modify the values of the parameters according to your needs.

Example:

```
curl --location --request POST 'https://web.sippo/sapi/permissions/user/{id}' \
--header 'Authorization: Bearer {bearerToken}' \
--header 'Content-Type: application/json' \
--data-raw '{
    "voicemail": {
        "enabled": true
    }
}'
```

Once sent, and if everything has worked correctly, you should receive as a response: 201 Created. In this case, they system will show the following log messages:

```
[/wac/lib/core/WIface.js] silly: HTTP POST /sapi/permissions/user/
↪5f9175bbf26dc8ba3a44f35d
[/wac/lib/services/OAuth2Provider/PassportStrategies.js] debug: token strategy, ...
↪40924497
[@quobis/db/lib/Storage.js] silly: search elements of collection oa_tokens with query, x.
↪token ==␣
↪3a0b39529a6057204dd6922e548bf3503ce0b7307a7513ad49c873a3864ce3f2b5c1ddc1f5d8e6a29ad63aa2212bc0079ebe136
[/wac/lib/core/Auth.js] silly: deserialize user with id 5ddcdb74bd53a58144f85915
[@quobis/db/lib/Storage.js] silly: search elements of collection user with query, ((x.id␣
↪== 5ddcdb74bd53a58144f85915))
[/wac/lib/core/Auth.js] silly: Authenticate success admin@quobis, [object Object]
[/wac/lib/services/Permissions/resources/ResourcePermissionsRestApi.js] silly: Received␣
↪an add permissions request
[@quobis/db/lib/Storage.js] silly: search elements of collection user with query, (x.id␣
↪== 5f9175bbf26dc8ba3a44f35d)
```

#### 7.4.3.2 Domain

You can activate voicemail at a domain level.

To do so use the POST request `Add Domain Permission`, that you can find in the WAC Permissions collection, followed by the domain id you want to modify. In the Body you must modify the values of the parameters according to your needs.

Example:

```
curl --location --request POST 'https://web.sippo/sapi/permissions/domain/{id}' \
--header 'Authorization: Bearer {bearerToken}' \
--header 'Content-Type: application/json' \
--data-raw '{
    "voicemail": {
        "enabled": true
    }
}'
```

Once sent, and if everything has worked correctly, you should receive as a response: `201 Created`. In this case, the system will log the following messages:

```
[/wac/lib/core/WIface.js] silly: HTTP POST /sapi/permissions/domain/
↪5ddcdb74bd53a58917f85914
[/wac/lib/services/OAuth2Provider/PassportStrategies.js] debug: token strategy, ...
↪40924497
[@quobis/db/lib/Storage.js] silly: search elements of collection oa_tokens with query, x.
↪token ==␣
↪3a0b39529a6057204dd6922e548bf3503ce0b7307a7513ad49c873a3864ce3f2b5c1ddc1f5d8e6a29ad63aa2212bc0079ebe136
[/wac/lib/core/Auth.js] silly: deserialize user with id 5ddcdb74bd53a58144f85915
[@quobis/db/lib/Storage.js] silly: search elements of collection user with query, ((x.id␣
↪== 5ddcdb74bd53a58144f85915))
[/wac/lib/core/Auth.js] silly: Authenticate success admin@quobis, [object Object]
[/wac/lib/services/Permissions/resources/ResourcePermissionsRestApi.js] silly: Received␣
↪an add permissions request
[@quobis/db/lib/Storage.js] silly: search elements of collection domain with query, (x.
↪id == 5ddcdb74bd53a58917f85914)
```

## 7.5 Early Media

Early Media is a feature supported by SIP calls that allows two SIP user agents to exchange media before the call is established. This media exchange usually is in the form of a ring tone that the caller hears. Some examples are in-band tones, progress information and other information messages related with queuing services or call status.

**Note:** More information about Early Media can be found in Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP) - tools.ietf.org

By default Early Media is always going to be supported in a SIP integration. This implies that when the SIP network supports Early Media and provides a ring tone it will be reproduced to the user. If the user web client also has a default ring tone for voice calls this means that the caller will hear a double ring tone, one from the Early Media and one from the ring file played by the web client. To avoid this overlap it is important to deactivate ring tone reproduced on the client side.

CHAPTER

EIGHT

# MEETINGS AND PUBLIC ROOMS

## 8.1 Meetings

Meetings are a special type of conferences with multiple participants, with the following characteristics:

- They have a URL is which know by every participant
- They have a start and end date
- They have a predefined list of attendees (participants)
- They can be created via a REST API

A meeting is accessed via a URL which is only valid for that specific meeting. This is URL is called the "meeting URL" and its starting path can be configured in the service configuration.

The meetings service allows users to schedule meetings where participants can be invited. The service will take care of sending a meeting invitation via email or SMS. For every email meeting invitation, an ICS (calendar) file is also sent as an attachment. This calendar file includes the meeting's name, its URL, organizer and start and end dates.

**Capabilities required**

Users that want to create and manage meetings must have the `meetings` capability assigned. If no meetings capability is assigned for a user, that user will be unable to schedule new meetings (but he still will be able to join someone else's meetings).

### 8.1.1 Configuring the service

The service is configured via the `wac.ini` file. It contains a top-level entry named `[meetings]` that holds the service's details.

| Parameter | Default value | Comment |
|---|---|---|
| `linkHost` | (blank) | (mandatory) The host that will be used to build invitation's links. If not given, the public IP of the Quobis AS will be used. |
| `inviteMechanisms[` | (blank) | Notification mechanisms used to inform the user about the meeting creation or update. Can be "email", "sms" or both. |
| `linkFormat` | (blank) | The format to use for the meeting URL. If `unique`, the meeting ID is used. If `user` the creator user's alias is used. |
| `templatePath` | meetings/tpl | Where to find the sms and email template folders, relative to `config/` |
| `shortenerToken` | (blank) | Needed token to short links by using a Bitly account. If left blank, this feature won't be used. |
| `languagesTemplate` | Available languages | Languages of the templates that will be used. Currently available ones are `es`, `en` and `ca`. By default, all will be used. |
| `remindBefore` | (blank) | If specified, the platform will send a reminder to every participant in the meeting `N` minutes before the start time. Additionally, a push notification is also sent, unconditionally |
| `pstnExtension` | (blank) | If specified, the meeting invitations will contain information in order to join the meeting through a regular phone call |

A configuration example of the meetings section in the `wac.ini` is shown below:

```
[meetings]
linkHost = https://meetings.wac/
inviteMechanisms[] = email
inviteMechanisms[] = sms
; templatePath = meetings/tpl
; shortenerToken = '<ACCESS TOKEN>'
; languagesTemplates[] = es
; languagesTemplates[] = en
; remindBefore = 10
; pstnExtension = +34999999999
```

### 8.1.2 Email notifications

When creating a meeting and specifying a valid email in the request, email notifications are sent to participants. These emails are sent using the optional `mailer` service. Templates are used in order to specify how the content of the email will look like. You can refer to the details reading the *Email service* documentation.

### 8.1.3 SMS notifications

When creating a meeting and specifying a valid phone (MSISDN format) in the participant information, SMS invites are sent to participants. Those SMS are sent using the optional `sendsms` service. To define how the invites look like, templates are used. You can refer to the details reading the *SMS notifications* documentation.

### 8.1.4 Mobile push notifications

Meetings makes use of the PushNotifications service for sending out a reminder of the meeting using mobile providers' push notifications service. The push payload will be:

```
{
    "type": "meeting-reminder",
    "info": {
            "id": meeting.id,
            "name": meeting.name
        }
}
```

Please note that above payload is not admin-configurable.

#### 8.1.4.1 Templates definition

One important aspect of the meetings service is the ability to configure the contents of the meeting notifications sent to participants. The service allows to configure them depending on the channel used to send them (SMS or email, currently) and the meeting's defined language. Templates are defined in the given `templatePath` from the configuration (relative to `./config`) and are structured in the following manner (default values used):

```
config/
 +- meetings/
     +- tpl/
         +- email/
         |   +- en/
         |   |   +- {from,html,subject,text}.hbs
         |   +- es/
         |       +- {from,html,subject,text}.hbs
         +- sms/
             +- en/
             |   +- {from,text}.hbs
             +- es/
                 +- {from,text}.hbs
```

As you can see, the file hierarchy is defined by the mechanism used to send the meeting invitation (sms or email) and, underneath, the language of the meeting. This way we have full control on how to build the meeting invitation. Additionally, for each of the mechanism, each of its parts is described in one single file for increased flexibility. For instance, in the case of the SMS, the sender of the SMS can be defined in the `from.hbs` file while the contents of the SMS itself, are defined in the `text.hbs` file. In the case of the email we have more fields available, so there are more files to be customized. Particularly for email we have both `text.hbs` and `html.hbs`. This way the email will be properly rendered in all email readers, no matter if they support HTML or not. All the templates are defined using `hbs` files, which are Handlebars files. Handlebars is a third party library that helps users to define templates by separating the JavaScript code from the HTML or text template itself. See the linked website above for further information.

Apart from the standard Handlebars tags and syntax, we provide the `formatDate` and `formatTime` helpers to localize the date strings to the meeting invitation's language. For example:

```
Your meeting {{ meeting.name }} will start on {{ formatDate
meeting.validSince }} at {{ formatTime meeting.validSince
hour="numeric" minute="numeric" hour12=false timeZone="Europe/Madrid" timeZoneName="short
↪" }}
```

You can see the full list of parameters that can be fed in the Intl documentation, under the `options` description. The data passed to the templates is the following:

| | |
|---|---|
| party | The email address or phone number of the recipient |
| meeting.name | Name of the meeting. |
| meeting.user.id | Id of the user creating the meeting. |
| meeting.user.domain | Domain of the user creating the meeting. |
| meeting.user.username | Username of the user creating the meeting. |
| meeting.user.email | Email address of the user creating the meeting. |
| meeting.user.created | Timestamp of when the user creating the meeting was created. |
| meeting.user.lastLogin | Timestamp of when the user creating the meeting last logged in. |
| meeting.user.role | Role of the user creating the meeting. |
| meeting.user.capabilities[] | Capabilities of the user creating the meeting. |
| meeting.user.mobilePhone | Mobile phone number of the user creating the meeting. |
| meeting.user.landLineNumber | Number of the user creating the meeting. |
| meeting.user.alias | Alias of the user creating the meeting. |
| meeting.users[] | Ids of the users participating in the meeting. |
| meeting.language | Default language for the meeting. |
| meeting.participants[] | List of the participants in the meeting. |
| meeting.validSince | Timestamp of the date since which the meeting is available. |
| meeting.validUntil | Timestamp of the date until which the meeting URL is available. |
| meeting.url | URL of the meeting. |
| meeting.ddi.extension | Meeting DDI extension. |

Please have a look at the provided templates in `config/meetings/tpl` to explore all the possibilities. A sample template may be:

```
<p>Hi {{ party }},<br/> you have been invited to a meeting via our communications␣
↪platform.</p>
<p>To join us, please
<a href="{{ meeting.url}}">this link</a>
{{#if meeting.ddi.extension}} or dial {{meeting.ddi.extension}} and introduce the code{
↪{meeting.ddi.pin}}{{/if}}.</p>
<p>If you cannot make it, please answer "No" to the RSVP.</p>
<p>Thanks!</p>
```

#### 8.1.4.2 Calendar file

When sending a meeting invitation by email, a calendar file is attached. This file is formatted according to RFC-5545. The main fields to note that are included are: meeting's starting and ending dates and times, meeting's name, URL and organizer. The organizer is an RFC-defined value of the format `Name <email@address.tld>`. This service will use the meeting's creator `username` as `Name` and its email address. If no email address is defined, a dummy `none@example.org` will be used.

#### 8.1.4.3 Logs: known log messages

**Error messages**

- `error: Invalid linkHost`: the configuration parameter `linkHost` must be set to a valid URL. Do not include a trailing slash.
- `error: Unable to send reminder for meeting: meetingId. Cause: msg`: an error occurred when trying to send a meeting reminder using Push Notifications. Please review Push Notifications service's logs.
- `error: Error while processing onSessionRegistered`: an error occurred when a user joined the meeting's URL. The full error must be printed following this line, which will give you details about the precise failure.
- `error: Error while processing onSessionDown`: an error occurred when a user disconnected from the meeting. The full error must be printed following this line, which will give you details about the precise failure.
- `error: Process onDeleteUser failed`: an error occurred when a deleted user was removed from its meetings. The full error must be printed following this line, which will give you details about the precise failure.

**Warning messages**

- `warn: unable to load resource for sms|email (msg)`: the service was unable to find the meeting invitation generator for the indicated mechanism. Probably the meeting invitation mechanism was misspelled or is not available.
- `warn: Cannot send the meeting invitation, not available`: a meeting invitation was not sent because the meeting created by the application was configured to do not send a meeting invitation from the sippo-server instance. This message is a confirmation that this server MUST NOT send a meeting invitation.
- `warn: Cannot send meeting invitation by email: Email service not available`: the meeting invitation cannot be sent by email because there's no such service enabled.
- `warn: Cannot send meeting invitation by sms: SMS service not available`: the meeting invitation cannot be sent by SMS because there's no such service enabled.
- `warn: Failed to invite participant INVITE to meeting NAME(id)`: the service was unable to send the given meeting invitation.
- `warn: The sms template TPL is too large. It will be sent several SMS by meeting`: the configured template, named `TPL` is too large to fit one single SMS. It will span across several SMS. This message only appears when the `languagesTemplates` variable is set.
- `warn: Field <FIELD> hasn't been used in TPL`: if an optional field `<FIELD>` available for the configured template `TPL` has not been used, this warning is issued. This message only appears when the `languagesTemplates` variable is set.

### 8.1.5 Related REST API endpoints

All of the following endpoints MUST be authenticated with the proper `Authorization` header.

- GET /sapi/meetings
- POST /sapi/meetings
- GET /sapi/meetings/:id
- PUT /sapi/meetings/:id
- DELETE /sapi/meetings/:id

## 8.2 Public rooms

Public rooms are conferences that are always available and that don't have a start or endtime. It's URL is created "on the fly" and there user can access the conference room even if they are not authenticated. Public rooms are reachable under the following path: http://your-system-domain/c/any-identifier. Any identifier that follows this pattern will result in the setup of a conference room that is reachable by any user who knows this URL.

**Warning:** Public rooms can be a quite handy resource but the system administrator needs to take into account that anyone can use the service.

CHAPTER

NINE

# MESSAGING

Quobis wac support text messaging between users by leveraging the XMPP standard protocol. Even thought XMPP is used for instant messaging, presence, multi-party chat, voice and video calls, collaboration and generalized routing of XML data, we are using XMPP just for instant messaging. The core XMPP technologies are defined in RFCs published by the IETF, mainly RFC 6120 (XMPP Core) and RFC 6121 (XMPP IM).

## 9.1 Architecture

There are two services to take into account: the `sippo-server.xmpp` service and the `xmpp-server` service. The `sippo-server.xmpp` service is an internal service of the `sippo-server` responsible of integrating the `xmpp-server` for:

- user authentication
- contact management
- active chat list management
- group chat features
- access control

## 9.2 Configuration

Initialize the service at `config/wac.ini` as with any other service:

```
; XMPP service
; This service enables communication with the XMPP server

; Configure it in `config/xmpp.toml`
[xmpp]
```

The file `config/xmpp.toml` needs to be configured as well with the following parameters:

- `xmppUrl`: URL to the WebSocket where the XMPP server is listening
- `hostsAdminUrl`: URL to the websocket where XMPP server will receive virtualhost updates
- `adminToken`: xmpp-server admin login token, set it to the same, random, and long-enough value as in prosody.cfg.lua

Example of `xmpp.toml` configuration:

```
#  - xmppUrl: the URL to the WebSocket where the XMPP server is listening
#  - hostsAdminUrl: the URL to the websocket where XMPP server will receive virtualhost␣
↪updates
```

(continues on next page)

(continued from previous page)

```
#  - adminToken: the Prosody's admin login token

[xmpp]
xmppUrl = "wss://web.sippo/xmpp-websocket"
hostsAdminUrl = "http://prosody:5280/quobis_virtualhosts/virtualhost"
adminToken = "please_change_me"
```

## 9.3 Messaging via REST-API

The sippo-server.xmpp service adds the following endpoint to the *REST API*. All of the following endpoints MUST be authenticated with the proper Authorization header:

```
GET /sapi/xmpp/login
PUT /sapi/xmpp/individualChatList/:jid
GET /sapi/xmpp/individualChatList
DELETE /sapi/xmpp/individualChatList/:jid
```

## 9.4 Capability implications

Users that need messaging features must have assigned the `chat` capability. If no `chat` capability is present for a user, that user will have all chat features disabled.

## 9.5 Troubleshooting

- Whenever messages are sent to an offline (or non-existing) user, those will be stored in the offline storage. If such user eventually logs in, all those offline messages will be forwarded to her. Take this into account if offline users are receiving lots of messages, user may be flooded with messages when becoming online.
- When debug logs are enabled at xmpp-server service will a huge amount information of information on the actions taking place. Activate only to report problems to engineering. Do not activate for production
- Please note that we only use XMPP for instant messaging, not for presence updates (we do have our own presence protocol)
- Please notice that currently standard XMPP clients are not supported, as not all the features would be available for them.

### 9.5.1 Logs: known log messages

The `xmpp` service will log messages according the globally configured log level of the `sippo-server`.

**Error level log messages**

- `error: Failed to obtain [admin] XMPP client`: an error occurred when trying to instantiate an XMPP client for communicating with `xmpp-server`. Please review connectivity and configuration parameter `adminToken` (and its counterpart in `prosody.cfg.lua`).
- `error: Failed to create <domain> host in XMPP server`: an error occurred when `POST``ing to ``hostsAdminUrl` for creating a new domain.

- `error: Authenticate no-user error Bearer realm="Users", error="invalid_token"`: an error occurred while authenticating a user (will also appear for other kind of login, non-XMPP ones). In order to discover which user failed to log in, you should check `xmpp-server` logs.

**Warning level log messages**

- `warn: Failed to send invite to MUC ${muc} to user ${user}`: unable to send invitation for a group chat to a user joining a group or a conference. Please review `xmpp-server` logs for additional information.
- `warn: Failed to revoke group membership of ${member} from ${muc}: ${error}`: unable to revoke membership of user leaving a group or leaving a conference.. Please review `xmpp-server` logs for additional information.

**Info level log messages**

No log messages are defined for this level.

**Debug level log messages**

- `debug: Kicking ${username} from ${muc}`: when a user has been removed from a group chat.
- `debug: Received a new usersGroup creation`: when a new group event is received
- `debug: Received a usersGroup deletion`: when a deleted group event is received
- `debug: Received a change in a group participants (joining)`: when a new participant enters a group event is received
- `debug: Received a change in a group participants (leaving)`: when a participant leaves a group event is received
- `debug: Received a notification of a new domain`: when a new domain is created event is received
- `debug: Received a notification of a deleted domain`: when a domain has been deleted event is received
- `debug: Received a notification of a new participant in a conference`: when a new participant joins a conference event is received
- `debug: Received a notification of a participant leaving a conference`: when a participant leaves a conference event is received
- `debug: Received a new conference`: when a new conference has been created event is received
- `debug: Received a conference destroyed`: when a conference has finished event is received

**Silly level log messages**

- `silly: Starting endpoints`: when SAPI endpoints are being setup
- `silly: Connecting to message-broker`: when the connection to RabbitMQ is being established.
- `silly: ConferenceCreated payload ${json}`: the payload received for a new conference event.
- `silly: Destroying MUC ${muc}`: when the MUC of a conference is going to be destroyed.

CHAPTER

TEN

# FILE TRANSFER

Quobis wac support the sharing of files between users of the platform. This functionality is also referred to as "file transfer" or "file upload" in the documentation. This service is not tied in any way to the *messaging service* as it does not uses XMPP protocol. Nevertheless, application developers usually include the file transfer functionality within the chat functionality, but this is not mandatory at all.

## 10.1 Configuration

This functionality is provided from the `sippo-server` service as an internal service and has some configuration options mostly for security reasons. In order to make use of it, administrators need to initialize the service at `config/wac.ini` as with any other `sippo-server` service.

These are the main parameters that need to be configured:

- `address`: the IP address to bind the service to. Defaults to 127.0.0.1
- `port`: the TCP port where the websocket will listen on. Defaults to 5678
- `uploadDir`: where to store temporary uploaded files. Defaults to /tmp/sfuploads, will be created if missing
- `chunkSize`: the size (in bytes) of the chunks a file will be splitted into. Defaults to 10240.
- `allowedExtensions`: array of string that refers mime type of allowed extensions. Default is empty array, which means "accept every file". Available since v4.1
- `maxFileSize`: bytes of max file size. Default value is 40MB. If undefined or zero, means no size limit. Available since v4.1
- `filenameLengthLimit`: max number of characters allowed. Default value is 40MB. Available since v4.1

An example of this configuration is shown below (please note that the section name is "fileupload"):

```
[fileupload]

address = 127.0.0.1
port = 5678
uploadDir = /tmp/sfuploads
chunkSize = 10240
allowedExtensions[] = image/jpeg
allowedExtensions[] = image/png
maxFileSize = 4194304
filenameLengthLimit = 40
```

If the user tries to upload a filetype which is not in the allowed extensions list, or that exceeds the maxFileSize or the filenameLenght, the application will thrown an specific error towards the SDK so the application developer can inform the end user.

CHAPTER

ELEVEN

# USER NOTIFICATIONS

The platform has the following ways of notifying internal and external users:

## 11.1 Email service

The platform provides a service for sending emails. That service may be used by other services that need to reach users via email.The emails to be sent can be created by the service's user or may use already defined templates. This allows the developer or the administrator to not need to define similar or equal emails more than once. Localized versions for such templates are also an option. All the features provided by the email service are supported by the underlying library nodemailer. Check it out if you need something that's not explicitly supported by this service.

### 11.1.1 Configuring the service

The service is configured via the `wac.ini` file as usual. It contains a top-level entry named `[mailer]` that holds the service's details. There is one mandatory option to be provided: `host`, which identifies the SMTP host:

**host**
: The host (name or IP address) of the SMTP server.

**username**
: The username for authenticating against the SMTP server.

**password**
: The password to use for the given username.

**port**
: The SMTP port to use (defaults to 587)

**secure**
: Whether to use TLS or not when talking with the SMTP. `false` still allow STARTTLS (defaults to false).

**pooled**
: Whether to use a connection pool or not (defaults to false)

**from**
: A forced From: address, ignoring each email's one (defaults to the one given in the email object)

**Note:** Currently, only SMTP/S is supported

### 11.1.2 Creating new templates

Templates use the handlebars syntax. It's an easy and intuitive replacement syntax. Just wrap your items to be replaced with `{{ }}` and you're ready to go. Please take a look at provided template and website to learn more about it.

To have a template ready to use, please drop it in a new folder inside `templatePath` (variable under `[meetings]` entry) . For instance, if you want to add a new template for sending emails for meetings, you can create a new folder called `email` inside `templatePath` and copy there your template files.

### 11.1.3 Template naming rules

Template must adhere to strict but simple naming rules. As the email service allows to send both plaintext and HTML versions of an email, there has to be a template for each of those versions. The plaintext one is mandatory and the HTML is optional. For the plaintext version, the file must be named `text.hbs`. The HTML template, is named `html.hbs`. If both templates are found, the email will have one alternate part for each of the versions. For example, this might be the real contents of `templatePath`:

```
templatePath/
`- email/
   `- html.hbs
   `- text.hbs
```

### 11.1.4 Localization of templates

There is also support for localization of templates. From the administrator point of view it is as easy as creating a subdirectory in the template directory named after the locale to be used. Inside this folder we can also include two more files `from.hbs` and `subject.hbs` which are copied to the corresponding fields of the email. That is, for our example `email` template we can provide a Spanish translation by creating a folder `es` inside it with the corresponding localized templates.

```
templatePath/
`- email/
   `- html.hbs
   `- text.hbs
   `- es/
      `- html.hbs
      `- text.hbs
      `- from.hbs
      `- subject.hbs
```

If no translation is requested, the `email` contents will be used.

**Note:** It's up to the service using the email service to ask for the localized versions of the email.

## 11.2 SMS notifications

The `sendsms` service provides the ability to send SMS to mobile devices. This service may be used by other services that need to reach users via SMS. The SMS will be created by already defined templates. These templates will include key words to be replaced by variable values. In addition there is the possibility to create different SMS texts based on the language selected. All the features provided by the sms service are supported by the underlying external service Beepsend. This external service has an API with which we can send SMS from the platform. To properly get this service working, it is necessary to create a profile on the provider platform. Check specific configurations on the corresponding backend configuration section.

### 11.2.1 Configuring the service

The service is configured via the `wac.ini` file as usual. It contains a top-level entry named `[sendsms]` that holds the service's details.

```
[sendsms]
backend[] = sendsms
; backend[] = linkmobility
```

**Optional paramters**

- `templatePath`: This key will define the path where to look for templates (defaults to `<rootdirectory>/config/smstemplates`).

It can be used as an absolute path or as a relative path. E.g.

```
templatePath = ./otherfolder
```

In order to find the templates in `<rootproject>/config/otherfolder`.

### 11.2.2 Available backends

- **sendsms** send SMS using Beepsend API. The backend will connect using a connection token with the API and will send one or several messages. When a message has more than 70 characters, Beepsend will create two related messages.

```
[backend.sendsms]
module = sippo-sms-bs
urlService = https://api.beepsend.com
token = xxxxxxxxxxxxxxxxxxxxxxxxx
from = Quobis
```

**Backend parameters**:

There are three mandatory options to be provided: `urlSevice`, `token` and `from`.

**urlService:**
The BeepSend API URL

**token:**
Connection Token for authentication.

**from:**
The sender id. It must be a string with one of the following formats:

- Alphanumeric. Maximum allowed characters are 11.
- MSISDN numbers or short numbers / codes. Between 9 and 17 chars.
- National format. Maximum length is 7 chars.

**encoding:**
Allows you to specify the message encoding . Available options are UTF-8, ISO-8859-15 or Unicode. Defaults to UTF-8.

**Link mobility** send SMS using the LinkMobility provider. To use it, enable it in the `wac.ini` file and set up the required parameters as follows:

```
[backend.linkmobility]
module = sippo-sms-linkmobility
login = username
password = <your_md5password>
from = QUOBIS
```

### 11.2.3 Creating new templates

Templates use the handlebars syntax. It's an easy and intuitive replacement syntax. Just wrap your items to be replaced with `{{ }}` and you're ready to go. Please take a look at provided template and website to learn more about it. To have a template ready to use, please drop it in a new folder inside `templatePath`. For instance, if you want to add a new template for sending SMS for meetings, you can create a new folder called `myMeetingsSMS` inside `templatePath` and copy there your template files.

### 11.2.4 Template naming rules

Template must adhere to a strict but simple naming rules. As the sms service sends only plaintext, the plaintext template is mandatory. The file must be named `text.hbs`. For example, this might be the real contents of `templatePath`:

```
templatePath/
`- myMeetingsSMS/
   `- text.hbs
```

### 11.2.5 Localization of templates

There is also support for localization of templates. From the administrator point of view is as easy as creating a subdirectory of the template directory named after the locale to be used. That is, for our example `myMeetingsSMS` template we can provide a Spanish translation by creating a folder `es` inside it with the corresponding localized templates:

```
templatePath/
`- myMeetingsSMS/
   `- text.hbs
   `- es/
      `- text.hbs
```

If no translation is requested, the `myMeetingsSMS` contents will be used.

### 11.2.6 Activity diagram

## 11.3 Mobile push notifications

Mobile push notifications are messages that pop-up on the user mobile devices. Push notifications do not require a particular app to be open on a device for the message to be received by the end user, so a smartphone user can see notifications even when their phone is locked or when the app is not running. Each mobile platform has support for push notifications, being iOS and Android, Fire OS the ones supported by Quobis wac.

**Note:** Please note that although there are lots of similarities, push notifications are not the same as text SMS messages.

### 11.3.1 Configuration

Quobis wac supports mobile pushes both for Android and iOS devices. It's required to add the push service to the wac.ini file as done with other services:

```
[pushnotifications]

# Configuration for Firebase - Android devices
firebase_key= replace_with_your_key  #key to authenticate against Firebase

# Configuration for APN - iOS devices
apn_key = replace_with_your_key           ; path to a .p8 file with the APN key
apn_keyId = replace_with_your_key_ID      ; key ID for the APN service
apn_teamId = replace_with_your_team_ID    ; team ID
apn_topic =  com.quobis.webco.voip        ; topic, for example com.quobis.webco.voip
apn_production = true                     ; true when the application is compiled for␣
↪production
```

### 11.3.2 Available logs

- Authorization config has neither token nor X509 information. Unable to continue.
- Unable to set up APNClient
- Configuring proxy as per "http_proxy" environment variable
- Unable to send push if no auth token is available.
- Refusing to send push without a token
- Message has no speficied type, setting it to . . .
- Will try to send this push notification: payload
- Failed to send push: result
- Refusing to send push without a token
- Error sending android push for token: token,

CHAPTER

TWELVE

# TROUBLESHOOTING

This section provides the information you need to maintain and troubleshoot the operation of the Quobis Communication Platform after it is installed, deployed and configured. The target audience for this guide comprises support or administration personnel who maintain a Quobis Communication Platform. This guide requires knowledge of telephony and IP networking technology.

This guide helps you perform the following tasks:

- Diagnose an issue by using the available tools to diagnose a problem.
- Locate the relevant information to identify the core problem by referencing to the relevant logs.
- Resolve issues faced by implementing the best troubleshooting practices.

## 12.1 Server side troubleshooting

### 12.1.1 Kubernetes troubleshooting

#### 12.1.1.1 Kubernetes cluster

##### Kubernetes API

Kubernetes stores the cluster state in the ETCD database among other metrics. This information can be obtained through an API. In order to access that data, you must run the commands explained in the following section in a host where the k8s certificates are configured to allow the access to the cluster API.

This section provides some troubleshooting ideas and solutions to common problems you might encounter with the Kubernetes cluster. The most relevant tool to learn here is the **kubectl** command line tool, which lets you control Kubernetes clusters. For configuration, kubectl looks for a file named config in the $HOME/.kube directory. You can specify other kubeconfig files by setting the KUBECONFIG environment variable or by setting the –kubeconfig flag. For more information about kubectl syntax, full command operations and details about each command, including all the supported flags and subcommands, see the kubectl reference documentation.

The first step should be checking the cluster API connectivity with the following command:

```
kubectl config view
```

In case that the API access is not set up in the VM where the command is executed, the output will be have no data:

```
quobis@ops-node1-delorean:~$ kubectl config view
apiVersion: v1
clusters: []
contexts: []
```

(continues on next page)

(continued from previous page)

```
current-context: ""
kind: Config
preferences: {}
users: []
```

The complete output must contain all the data related with selected cluster:

```
quobis@OPS-master-delorean:~$  kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://IP:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    namespace: dev-nightly
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

Make sure the selected context are the cluster we what to manage. The row with an asterisk in the beginning defines the current context and the last column defines the default namespace selected ("my-context" and "dev-nightly" in the example below):

```
console:~/Devops/k8s-installer/ansible$ kubectl config get-contexts

        CURRENT NAME             CLUSTER                    AUTHINFO                 NAMESPACE
            akspreGDQuobis           akspreGDQuobis                clusterUser_preGDQuobis_akspreGDQuobis                 quobis
            aksproGDQuobis           aksproGDQuobis                clusterUser_proGDQuobis_aksproGDQuobis                 wac-quobis
            aksstaging               aksstaging                    clusterUser_testGDQuobis_aksstaging                    stagingquobis
            cluster.quobis.com       cluster.quobis.com            cluster.quobis.com                                     quobis-qa
            clusterquobis.quobis.com clusterquobis.quobis.com   clusterquobis.quobis.com                                kube-system
 *          my-context               kubernetes                    kubernetes-admin                                       dev-nightly
            preprod                  preprod                       clusterUser_test_preprod                                quobis
            proGD                    proGD                         clusterUser_pro_gemelo_digital_proGD                    wac-quobis
```

In order to change to another context, use the following command:

```
kubectl config use-context <desired-context>
```

In order to change the default namespace, use the following command:

```
kubectl config set-context --current --namespace=<desired-namespace>
```

### Kubernetes cluster state

The main element in a Kubernetes cluster are the nodes (VM) where containers are running, so checking the nodes' state should be the first step once the k8s API is configured. The cluster node must have a "Ready" state, any other case indicates one of the problems.

```
quobis@OPS-master-delorean:~$ kubectl get nodes -o wide
NAME                        STATUS   ROLES    AGE    VERSION   INTERNAL-IP   EXTERNAL-IP␣
↪  OS-IMAGE                          KERNEL-VERSION   CONTAINER-RUNTIME
k8s-master                  Ready    master   140d   v1.15.5   10.1.72.104   <none>     ␣
↪      Debian GNU/Linux 9 (stretch)   4.9.0-16-amd64   docker://18.9.9
kubernetes-janus-edu        NotReady <none>   140d   v1.15.5   10.1.72.107   <none>     ␣
↪      Debian GNU/Linux 9 (stretch)   4.9.0-8-amd64    docker://18.9.9
ops-k3s                     Ready    <none>   107d   v1.15.5   10.1.72.71    <none>     ␣
↪      Debian GNU/Linux 9 (stretch)   4.9.0-9-amd64    docker://18.9.9
ops-kubernetes-janus2-edu   Ready    <none>   140d   v1.15.5   10.1.72.108   <none>     ␣
↪      Debian GNU/Linux 9 (stretch)   4.9.0-9-amd64    docker://18.9.9
ops-node1-delorean          Ready    <none>   140d   v1.15.5   10.1.72.105   <none>     ␣
↪      Debian GNU/Linux 9 (stretch)   4.9.0-12-amd64   docker://18.9.9
ops-node2-delorean          Ready    <none>   140d   v1.15.5   10.1.72.106   <none>     ␣
↪      Debian GNU/Linux 9 (stretch)   4.9.0-9-amd64    docker://18.9.9
```

### Kubernetes node troubleshooting

```
Kubectl describe node <unhealth_node_name>
```

- **Check the state of VM: the Virtual Machine must accomplish the following requirements:**
  - It must be on.
  - It must have network connectivity to the rest of the cluster machines.
- **Check kubernetes dependencies running on the VM:**
  - Kubelet service must be running: `sudo systemctl status kubelet`
  - Kubelet logs: `sudo journalctl -u kubelet` (error logs should not appear)
  - The iptables command must contain the following entries (`sudo iptables -L`), several entries can insert additional delays in the system.
- Check the native k8s pod in kube-system namespace:

```
kubectl -n kube-system get pods
```

- Make sure `kube-flannel` is deployed and running.

In order to check the state of the pods of a given namespace pod, we need to use the following command:

```
kubectl get pods -n namespace_name
```

In order to get additional information of a specific pod, the instruction to issue is the following one:

$ kubectl -n <namespace> describe pod <pod_name>

In order to reboot a pod, we just need to delete it and Kubernetes will recreate it again. The command to issue is:

$ kubectl -n <namespace> delete pod <pod_name>

### Kubernetes namespaces and pod list

The following namespaces are always available in our platform after a clean installation:

- **"kube-system"** namespace: used internally by Kubernetes.
- **"monitoring"** namespace: host the monitoring services (Grafana, Loki, etc)
- **"quobis"** namespace: hosts the application services (authentication server, signalling server, databases, etc..)

This is the list of the pods that run in the **kube-system** namespace, that can be listed with the `kubectl get pods -n kube-system`:

```
quobis@vm-bar-quo-mas11-1:~$ kubectl get pods -n monitoring
NAME                          READY STATUS  RESTARTS AGE
controller-6c4449fc67-pfn48   1/1   Running 1        55d
coredns-5c98db65d4-9r99t      1/1   Running 1        55d
coredns-5c98db65d4-l8v4v      1/1   Running 1        55d
etcd-vm-bar-quo-mas11-1       1/1   Running 9        55d
kube-apiserver-vm             1/1   Running 10       55d
kube-controller-manager-vm    1/1   Running 18       55d
kube-flannel-ds-2z2qm         1/1   Running 0        55d
kube-proxy-9mflq              1/1   Running 0        55d
kube-scheduler-vm-bar         1/1   Running 14       55d
nginx-ingress-controller      1/1   Running 0        22d
node-exporter-6xm5h           1/1   Running 1        55d
speaker-cxf8r                 1/1   Running 1        55d
```

**Note:**

- There might be more that one pod with the same name but different suffix (for example, `coredns-5c98db65d4-9r99t` and `5c98db65d4-l8v4v`). These are just replicas of the same image running at the same time into this namespace.
- You might not find all these services in your deployment, as some of them are optional and/or might be deactivated. For example, the "teal" service is not deployed if the deployment does not require blue/green support.

This is the list of the pods that run in the **monitoring** namespace, that can be listed with the `kubectl get pods -n monitoring`:

```
quobis@vm-bar-quo-mas11-1:~$ kubectl get pods -n monitoring
NAME                          READY  STATUS    RESTARTS AGE
grafana-54d5f96758-m4gtj      1/1    Running   0        27d
loki-5d8c457556-8qdhb         1/1    Running   0        51d
prometheus-b9d4f8f54-dpxtv    1/1    Running   0        21d
promtail-c626r                1/1    Running   0        20d
```

Table 1: Pods included in the monitoring namespace

| Pod name | Functionality | Deployment considerations |
|---|---|---|
| `grafana` | Grafana instance for logging and troubleshooting. | No HA required. |
| `loki` | Loki instance for log collection and management. | No HA required. |
| `prometheus` | Prometheus instance as a storage for Grafana and Loki | No HA required. |
| `promtail` | Agent which ships the contents of local logs to a the Grafana Loki instance | No HA required. |

This is the list of the pods that run in the **quobis** namespace, that can be listed with the `kubectl get pods -n quobis`:

```
quobis@vm-bar-quo-mas11-1:~$ kubectl get pods -n quobis
NAME                                        READY   STATUS  RESTARTS AGE
audiomixer-sfu1-0                           1/1     Running 0        22d
audiomixer-sfu2-0                           1/1     Running 0        22d
database-mongo-0                            2/2     Running 0        6d4h
database-mongo-1                            2/2     Running 0        6d4h
database-mongo-2                            2/2     Running 0        6d4h
erebus-787567f497-lnrcr                     1/1     Running 0        6d15h
kapi-759d58f644-m5db2                       2/2     Running 0        22d
keycloak-77f6796b89-9ngw7                   1/1     Running 0        6d16h
kitter-5b5647c888-dbdfw                     1/1     Running 0        13d
kube-state-metrics-bc95796c9-hb42c          2/2     Running 0        43d
message-broker-5675fb65b9-755s6             1/1     Running 0        6d16h
mongodb-exporter-548d4dcddd-7bt67           1/1     Running 0        43d
mongodb-kubernetes-operator-6c6cbdf9bf-8jwg 1/1     Running 0        6d16h
nfs-client-provisioner-5697c964d6-pwgl5     1/1     Running 0        6d16h
nginx-gw-0                                  1/1     Running 0        13d
podhandler-1640265360-4n5vw                 0/1     Running 0        13d
postgresql-5c9f585dc8-bw5t9                 1/1     Running 0        6d16h
qss-audiomixersio-sfu1-86cf778b84-28ct2     1/1     Running 0        6d15h
qss-audiomixersio-sfu2-57454f59d-brswp      1/1     Running 0        6d15h
qss-auth-http-679d4475d7-w65ch              1/1     Running 0        6d15h
qss-calls-574cbbbcdd-j94wx                  1/1     Running 0        6d15h
qss-calltransfer-basic-5c66bf9764-lg2vk     1/1     Running 0        6d15h
qss-conference-state-7b8d4f448-dnbhk        1/1     Running 0        6d15h
qss-invites-rooms-7d7749896f-rhbjw          1/1     Running 0        6d15h
qss-io-websockets-6bcd86694c-pzd8p          1/1     Running 0        6d15h
qss-io-websockets-6bcd86694c-rv8fq          1/1     Running 0        6d15h
qss-log-conference-7fdd67d9f5-gkdrt         1/1     Running 0        6d15h
qss-meeting-basic-6d6789b86-h7642           1/1     Running 0        6d15h
qss-peer-jt-6468689c87-7t5hg                1/1     Running 0        6d15h
qss-quick-conference-844d87c4f5-4v8j8       1/1     Running 0        6d15h
qss-registry-authenticated-59dc88c8ff-b6w4d 1/1     Running 0        3h14m
qss-resolver-wac-6f4787c4f4-9fcv5           1/1     Running 0        6d15h
qss-rooms-basic-6798d49d48-lmf9v            1/1     Running 0        6d15h
qss-trunk-85dcc5cb9f-jd87x                  1/1     Running 0        3h13m
qss-watchdog-invites-6dc97486f6-qstg2       1/1     Running 0        6d15h
qss-watchdog-registry-675f8b55c9-58bm7      1/1     Running 0        6d15h
sfu-dispatcher-66f58bf667-bll57             1/1     Running 0        8d
sfu-wrapper-sfu1-598d59fd79-mplxb           1/1     Running 4        8d
sfu1-0                                      1/1     Running 0        22d
sfu2-0                                      1/1     Running 0        22d
```

(continues on next page)

(continued from previous page)

```
sip-proxy-1-0                              1/1     Running 0        22d
sip-proxy-2-0                              1/1     Running 0        22d
sip-proxy-database-6bf67bc969-q4v2j        1/1     Running 1        55d
sippo-exporter-678ff77cff-4dztt            1/1     Running 0        14d
sippo-maintainer-1640228400-tldsm          0/1     Completed        0d10h
sippo-server-755f6d8f9b-ckgzx              1/1     Running 7        22d
sippo-server-755f6d8f9b-pzmdh              1/1     Running 0        6d15h
sippo-server-755f6d8f9b-q7gmr              1/1     Running 7        22d
sippo-storage-86c7bcb6d4-bhbnf             1/1     Running 0        6d16h
webphone-angular-8677ccbcc8-gkmh9          1/1     Running 0        12d
xmpp-server-58ccd4bb86-b6s4t               1/1     Running 0        41d
```

The list below gives a brief explanation of each service and its impact in case or an issue with this pod.

Table 2: Pods included in quobis namespace

| Pod name | Functionality | Deployment considerations |
|---|---|---|
| `audiomixer` | Audio *MCU*, handles the mixing of the audio streams and media interconnection with the PSTN/NGN. | Deployed in HA. If one instance gets down, all the active calls using it will fail. Following calls will use the active instance. |
| `database-mongo` | Internal database to storage user data. | Deployed in HA as a cluster. |
| `erebus` | Responsible for listening to WebSocket connections from clients and feeding them into our internal message broker | No HA, needs to be re-created in case of failure as all the clients traffic goes through this element |
| `kapi` | Provides a REST API used for the Kubernetes cluster maintenance. | HA not required. No critical impact in the service in case of failure. |
| `keycloack` | Identity and access management solution | No HA. Needs to be re-created in case of failure, otherwise users without a session token won't be able to login. |
| `kitter` | Provides an API to manage user accounts (CRUD). | No HA. Needs to be re-created in case of failure, otherwise no news users can be added to the system. |
| `message-broker` | Message broker using AMQP protocol based on RabbitMQ | No HA. Needs to be re-created in case of failure, otherwise there won't be message interchange between several services. |
| `mongodb-exporter` | Exports data from the Mongo database to the monitoring system | No HA. No impact in the service in case of failure. |
| `mongodb-kubernete` | Manages the lifecycle events for a MongoDB cluster by using the K8S API | No HA. No impact in the service in case of failure. |
| `nfs-client-provis` | Atomatic provisioner that uses existing NFS server to support dynamic provisioning of Kubernetes Persistent Volumes. | No HA. No impact in the service in case of failure. |
| `nginx` | Web server, reverse proxying, caching and load balancing for HTTP, TCP, and UDP servers. | HA provided by the cluster. |
| `postgresql` | Postgresql database. | No HA. Used by the chat messaging server, voice and audio calls are not affected by a potential failure. |
| `qss` | Quobis Signaling Server | All QSS microservices can be replicated. More info *here*. |
| `recording` | Responsible for post-call processing to create audio and video recordings. | No HA, needs to be re-created in case of failure as calls won't be recorded. |
| `sfu-dispatcher` | Balances the traffic between different video SFUs | No HA, needs to be re-created in case of failure. The call information is kept in the DB to be persistent across restarts. |
| `sfu-wrapper` | Connector between the SFU dispatcher and the SFU itself | Supports HA. If one instance gets down, all the active calls using it will fail. Following calls will use the active instance. |
| `sfu` | Video *SFU*. | Supports HA that is managed by the SFU wrapper and dispatcher services. |
| `sip-proxy` | *SIP application server* that acts as a gateway to an external telephony network | Supports HA. |
| `sip-proxy-databas` | MySQL database backend for the SIP proxy. Supports HA if required (not deployed by default) | Supports HA if required (not deployed by default) |
| `sippo-exporter` | Exports aggregated data to the monitoring system. | No HA needed. |
| `sippo-maintainer` | Runs daily operations for system maintenance. | No HA needed. |
| `sippo-server` | Main process that orchestrates the sytem. | HA supported, several replicas can be deployed and traffic is balanced from the message broker. |

## 12.1.2 Database

### 12.1.2.1 Introduction

The Quobis Communication platform uses a MongoDB database to store the information. The official documentation can be found here https://docs.mongodb.com/manual/.

### 12.1.2.2 Connection policy

The connection with the MongoDB database is handled by a library shared between all the services in the platform, so all of the uses the same mechanisms to connect, reconnect and disconnect from the database. In case that the connection with MongoDB doesn't work, the following error is logged:

```
[error] No connection with MongoDB. Check its URI or status (attempt: 1);
```

The library has a reconnection mechanism that tries to connect up to 30 times at a rate of one per second. These values match the defaults in MongoDB driver and cannot be configured. The same mechanism is executed not only at start time but also in case of a disconnection. In case that the connection is not successful after these attempts, the following error is logged and no further connection attempts are made:

```
[error] Failed to reach MongoDB: reconnection retries exhausted (30). Please fix the␣
↪error and retry.
```

## 12.1.3 Log analysis

This chapter describes the logs available with the Quobis Communication Platform and explains how to access and view them. It also explains the relationship between logs and system events. In order to get logs form the pods deployed into the Kubernetes cluster, we use the following command:

```
$ kubectl logs -f $Service-name > log-$servicename.log
```

On the other hand, there is a Loki service available that is integrated into Grafana for a better log collection experience.

Fig. 1: Log view and inspection with Loki tool within Grafana

We can find logs by string in order to trace an event or make troubleshooting easier.

To explore further, go to the Json and Data tabs. Here you can inspect the payloads and the responses of the logged data sent to Loki. Below, we can see options of how we can download the logs from Grafana from the Data tab on Grafana Loki. The options provided by Grafana Loki are to download the data as a CSV file or as a log file that can then be shared with other logging platforms if needed.

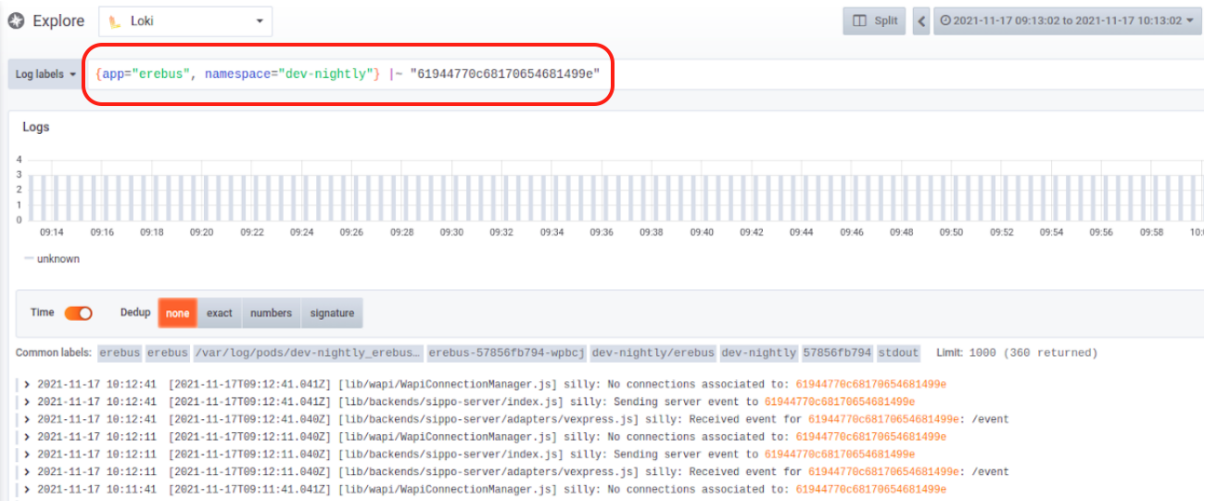



Fig. 2: Example of searching and filtering

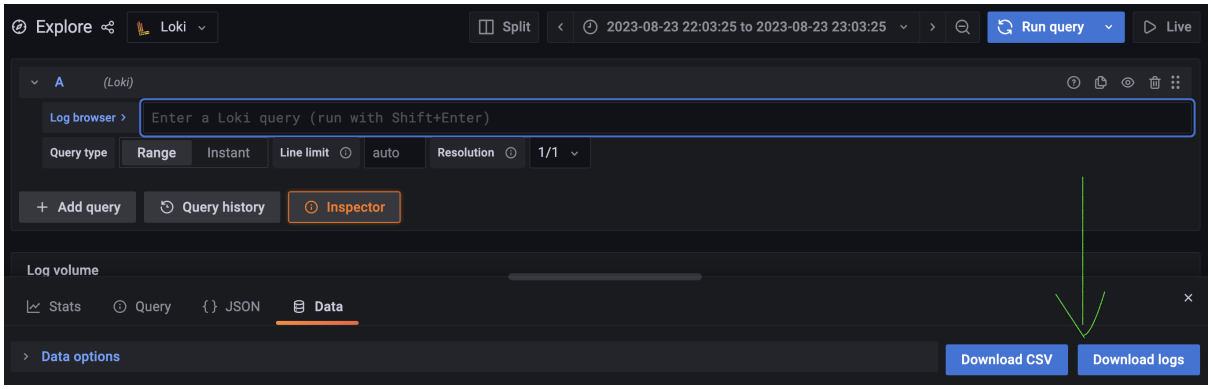



Fig. 3: Where to download logs in CSV and TXT

#### 12.1.3.1 Log levels

Each log message has an associated log level that gives a rough guide to the importance and urgency of the message. These are the available log levels in the platform, from more severe to less severe:

Table 3: Available log levels

| Level name | Level description |
|---|---|
| 'error' | Error events of considerable importance that will prevent normal program execution, but might still allow the application to continue running. |
| 'warning' | Potentially harmful situations of interest to end users or system managers that indicate potential problems. |
| 'info' | Informational messages that might make sense to end users and system administrators, and highlight the progress of the application. |
| 'verbose' | Informational messages with more detailed information |
| 'debug' | Information broadly interesting to developers, might include minor (recoverable) failures and issues indicating potential performance problems. |
| 'silly' | The most detailed log messages, may contain sensitive application data. |

Each service can have a different log level, the default configuration is as follows:

Table 4: Default log levels

| Service name | Log level |
|---|---|
| `audiomixer` | info |
| `database-mongo` | info |
| `erebus` | silly |
| `kapi` | info |
| `keycloack` | debug |
| `kitter` | silly |
| `message-broker` | info |
| `mongodb-exporter` | error |
| `mongodb-kubernetes-operator` | debug |
| `nfs-client-provisioner` | info |
| `nginx` | debug |
| `postgresql` | warning |
| `qss` | debug |
| `recording` | warning |
| `sfu-dispatcher` | debug |
| `sfu-wrapper` | debug |
| `sfu` | info |
| `sip-proxy` | info |
| `sip-proxy-database` | warning |
| `sippo-exporter` | info |
| `sippo-maintainer` | info |
| `sippo-server` | debug |
| `sippo-storage` | warning |
| `teal` | warning |
| `turn-server` | warning |
| `xmpp-server` | debug |

### 12.1.4 Monitoring

Monitoring has been built with the opensource tool Grafana. The URL of the Grafana access is https://base_url:/grafana and the credentials must be provided by Quobis support. It shows different dashboards with the live metric/logs values and metric history. By default the following dashboards are included:

In order to provide a first check of the system, the Kubernetes cluster monitoring dashboard where the resources of the cluster is shown. Below you can see a screenshot of a running system during five working days:

This is another screenshot during a normal day, where we can see the correlation between connected users, concurrent conferences and the network bandwith:

It also provides the alerting service, where the following alerts are included by default in the deployment:

- VM CPU/RAM/DISK usage.
- Internal traffic load (inside the cluster)
- Pods and container CPU/RAM/DISK usage.
- Database status

Several alerts can be added in function on the needs of the deployment. The following image describes the configuration of an alert:

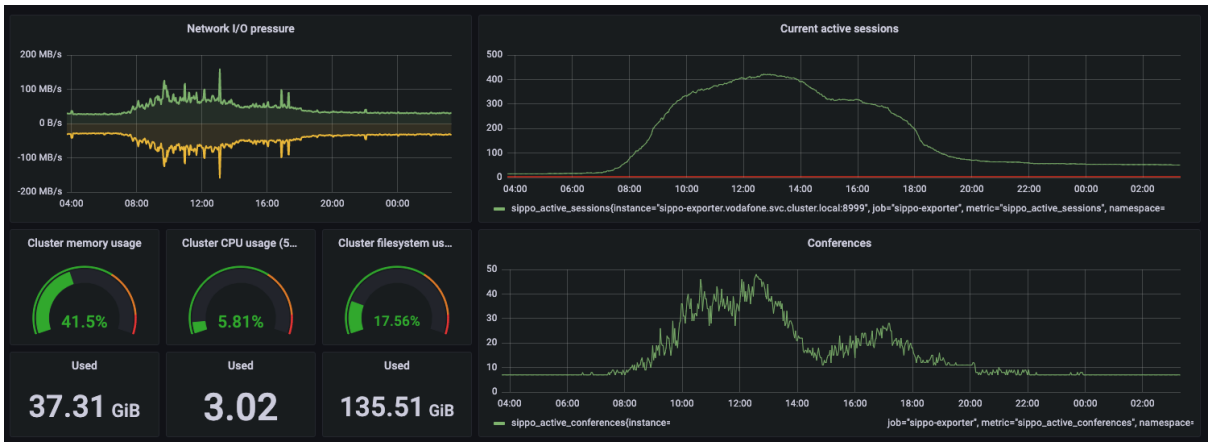


Fig. 4: Example of alerts fired in the system

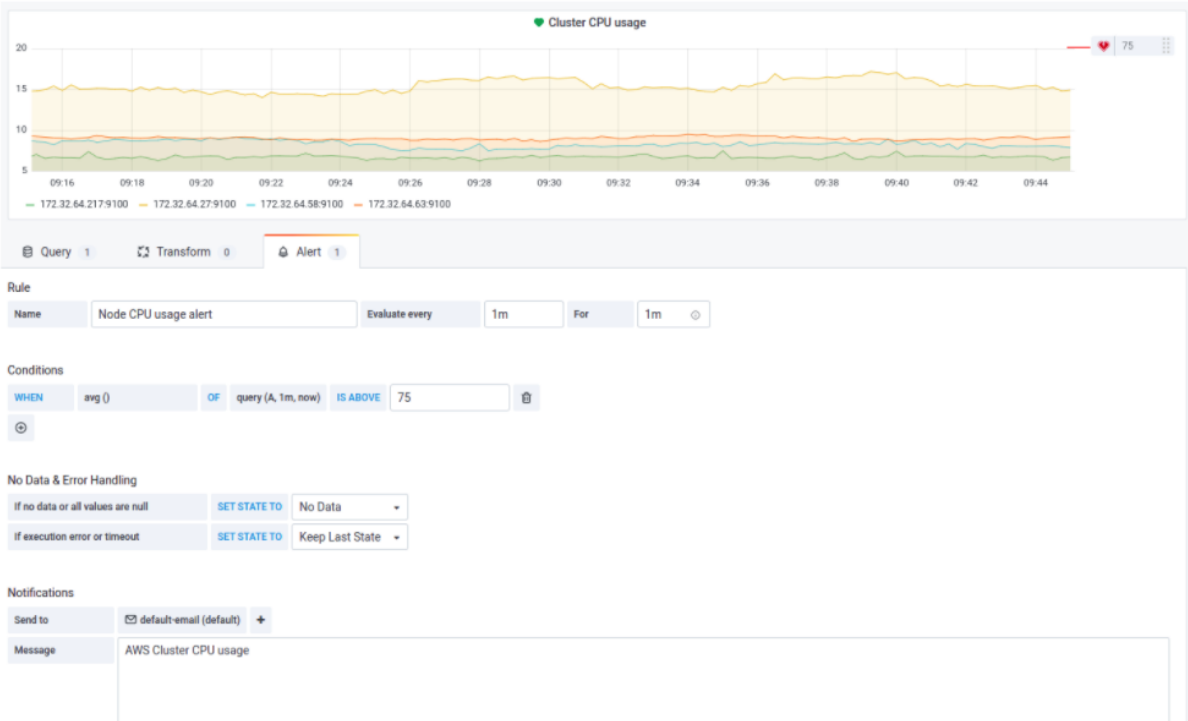
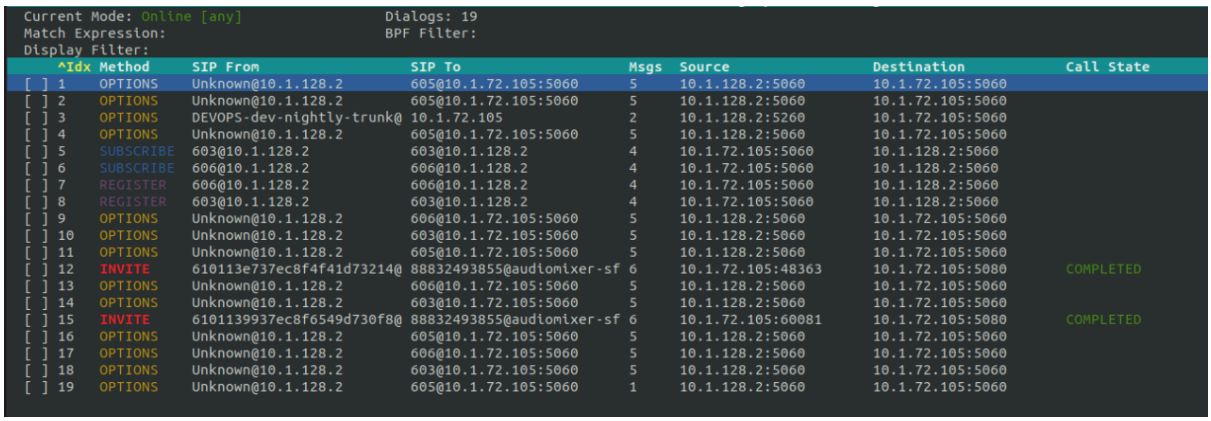



Fig. 5: Example of the alert configuration

### 12.1.5 Call Tracing

There are several tools available to trace a call event:

- **Asterisk: we can access the Audiomixer container and execute the following commands in order to capture all the events, and even, check all the available channels in the call:**
  - kubectl exec -it <asterisk_pod_name> bash
  - asterisk -rvvvvvvv
  - pjsip show channels (expected result)
- QSS audiomixer logs: its logs show all the communication between the QSS and Asterisk Manager Interfaces.
- "Sngrep" tool: run sngrep command in the media server in order to trace all SIP messages.



```
Current Mode: Online [any]                    Dialogs: 19
Match Expression:                             BPF Filter:
Display Filter:
    ^Idx Method     SIP From                  SIP To                    Msgs  Source              Destination         Call State
[ ] 1    OPTIONS    Unknown@10.1.128.2        605@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 2    OPTIONS    Unknown@10.1.128.2        605@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 3    OPTIONS    DEVOPS-dev-nightly-trunk@ 10.1.72.105               2     10.1.128.2:5260     10.1.72.105:5060
[ ] 4    OPTIONS    Unknown@10.1.128.2        605@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 5    SUBSCRIBE  603@10.1.128.2            603@10.1.128.2            4     10.1.72.105:5060    10.1.128.2:5060
[ ] 6    SUBSCRIBE  606@10.1.128.2            606@10.1.128.2            4     10.1.72.105:5060    10.1.128.2:5060
[ ] 7    REGISTER   606@10.1.128.2            606@10.1.128.2            4     10.1.72.105:5060    10.1.128.2:5060
[ ] 8    REGISTER   603@10.1.128.2            603@10.1.128.2            4     10.1.72.105:5060    10.1.128.2:5060
[ ] 9    OPTIONS    Unknown@10.1.128.2        606@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 10   OPTIONS    Unknown@10.1.128.2        603@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 11   OPTIONS    Unknown@10.1.128.2        605@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 12   INVITE     610113e737ec8f4f41d73214@ 88832493855@audiomixer-sf 6     10.1.72.105:48363   10.1.72.105:5080    COMPLETED
[ ] 13   OPTIONS    Unknown@10.1.128.2        606@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 14   OPTIONS    Unknown@10.1.128.2        603@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 15   INVITE     6101139937ec8f6549d730f8@ 88832493855@audiomixer-sf 6     10.1.72.105:60081   10.1.72.105:5080    COMPLETED
[ ] 16   OPTIONS    Unknown@10.1.128.2        605@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 17   OPTIONS    Unknown@10.1.128.2        606@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 18   OPTIONS    Unknown@10.1.128.2        603@10.1.72.105:5060      5     10.1.128.2:5060     10.1.72.105:5060
[ ] 19   OPTIONS    Unknown@10.1.128.2        605@10.1.72.105:5060      1     10.1.128.2:5060     10.1.72.105:5060
```

Fig. 6: Example of a SIP capture with SNGREP showing a list of dialogs

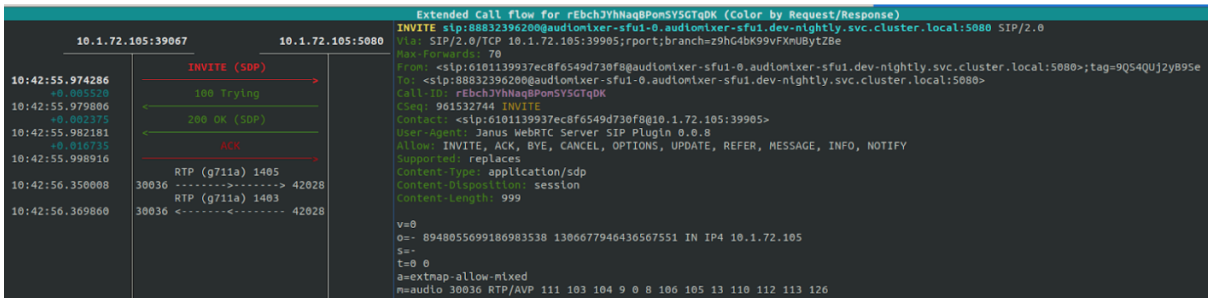



Fig. 7: Example of a SIP capture with SNGREP showing the extended call flow and ladder diagram

## 12.2 Client troubleshooting

Some issues can be specific to a user, and may be difficult to troubleshoot without more detailed information. The developer console is a tool which logs the information associated with a web page, such as JavaScript, network requests, and security errors. Console logs help debug issues by giving insights into error messages that get documented by your browser.

If your issue is still happening, please help us capture the console logs for the error by performing the following steps. The following are instructions for how to open the developer console on various different browsers.

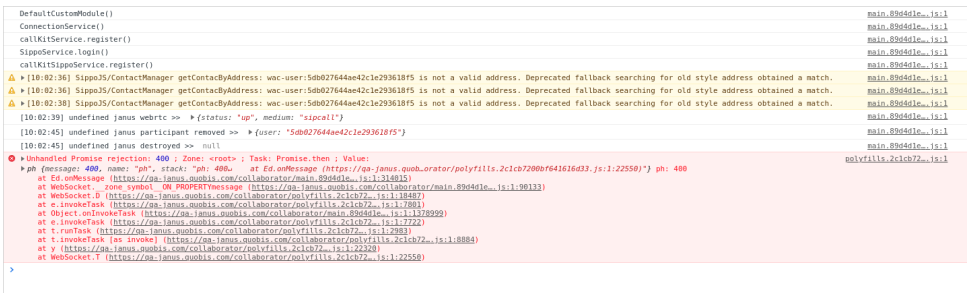
Once you have located the console, please send us a screenshot or export its contents to a plain log file using the "export" or "Save As" options (right-click).

### 12.2.1 Getting logs with the browser

#### 12.2.1.1 Showing the console

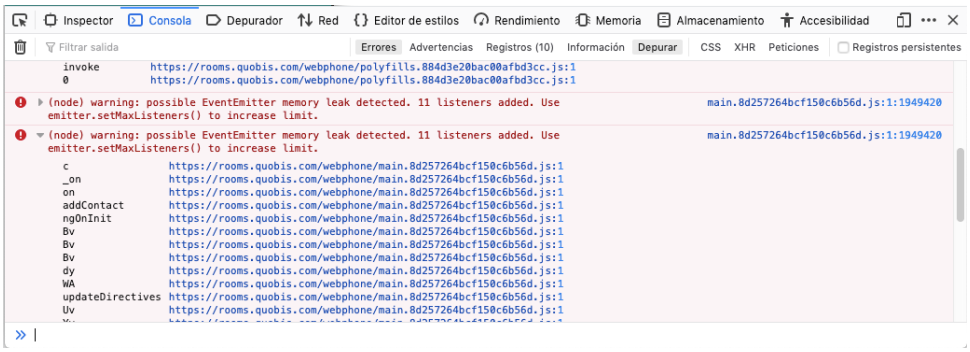
To show the console:

- Click on Chrome settings button (the three dots in the upper-right hand corner of the browser)
- Go to **More tools** > **Developer tools**. This should open up the Chrome developer tools panel.
- Alternatively, you can use the keyboard shortcut **Cmd+Shift+J** (on Windows) or **Cmd+Option+J** (on Mac).
- Click on the "Console" tab in the Chrome developer tools panel:



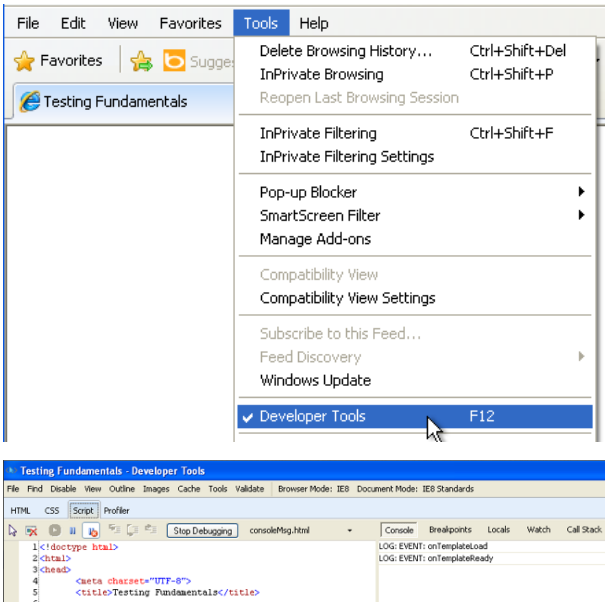
### Firefox

To open the console on Firefox, use the keyboard shortcut **Cmd+Shift+K** (on Windows) or **Cmd+Option+K** (on Mac). The toolbox will appear at the bottom of the browser window, with the Web Console activated.



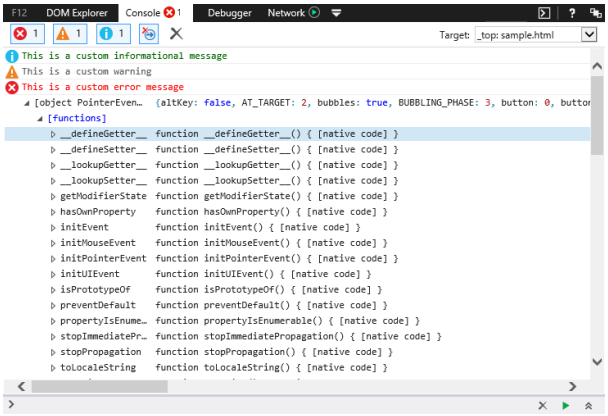

### Internet Explorer

To view the console messages, open the Developer Tools. In the Internet Explorer menu bar, select Tools > Developer Tools, or press F12. IE Developer Tools In the Developer Tools window, select the Script menu item and you should see your console messages displayed.



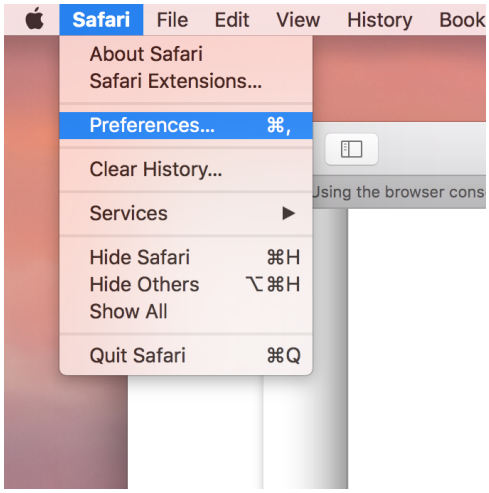
### Microsoft Edge

To open the console on Edge, hit F12 to access the F12 Developer Tools. Once in the F12 Developer Tools, navigate to the Console tab.

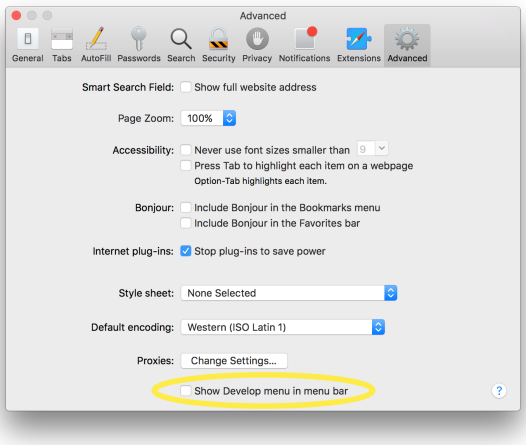



### Safari

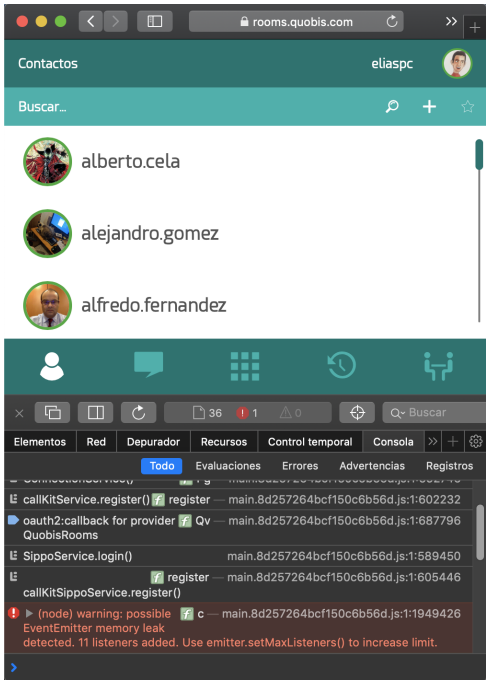
To open the console on Safari, you will first need to turn on the Develop menu. To do this, open the Safari menu in the Mac menu bar, then select Preferences.




Once in the Preferences dialog, navigate to the Advanced tab, then check the **"Show Develop menu in the menu bar"** box.



Once the Develop menu has been enabled, you can access it in the menu bar and then select the **"Show JavaScript Console"** option. The JavaScript Console will show up on the bottom half of the active browser window. Alternatively, you can use the keyboard shortcut **Option+Cmd+C**, though note that you will need to enable the Develop menu in your Safari Preferences for this shortcut to work.




### 12.2.1.2 Dumping WebRTC-related logs to file

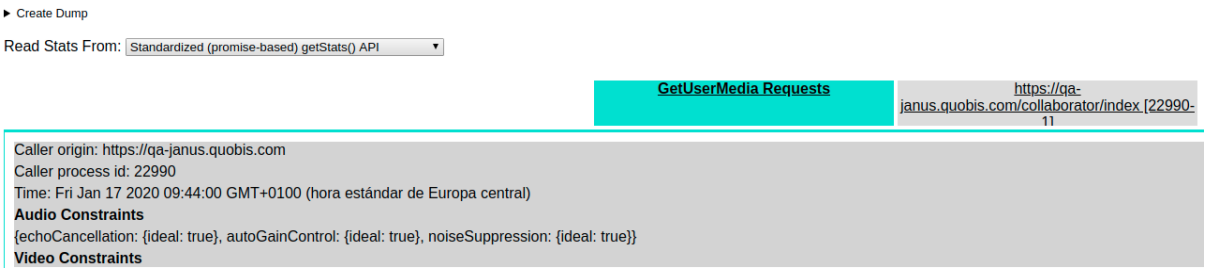
WebRTC logs help developers and technical support in case of issues with video and audio connections. Right now only Chrome and Firefox web browser allow the user to access those logs.

The following are instructions for obtaining on both browsers the logs and downloading them to a file.

#### Chrome

First of all you should restart the browser as it keeps track of all WebRTC connections. Once restarted, open a new tab and point to "**chrome://webrtc-internals/**" URL. You should see something like this:

Now open on a new tab your application and reproduce the problem. On the webrtc-internals tab should have appeared two new tabs as shown below.

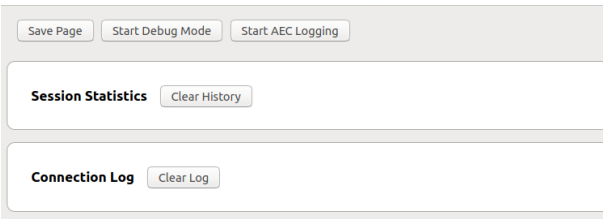



Select the one with the name of the application and click the **Download the PeerConnection updates and stats data** button.

After downloading, you will obtain a plain text file (extension TXT), remember to attach it to your Quobis support ticket. For debugging purposes, you can use your favorite editor to examine the content.

#### Firefox

For the same reason as in Chrome you should firstly restart your browser. Then, open a new tab pointing to "**about:webrtc**".



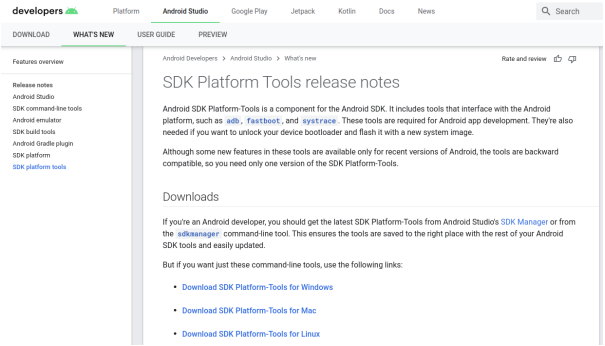

Now open the application and reproduce the problem. Go back, without closing the application, to the "**about:webrtc**" tab and click **Save Page**.

After downloading, you will obtain an HTML file, which can be visualized in any browser. Remember to attach it to your Quobis support ticket.

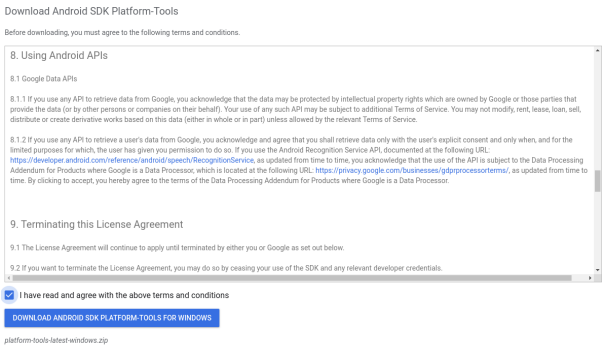
### 12.2.2 Getting logs from mobile applications

#### 12.2.2.1 Android applications

First of all, download the platform SDK tools for the operating system the computer has from the following link: https://developer.android.com/studio/releases/platform-tools.



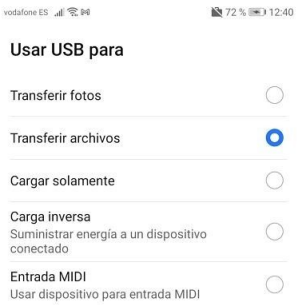

Select the appropriate version and agree to the terms and conditions.




Then, extract the compressed file to a directory of your choice. The extracted folder contains a `logcat`. Logcat is a command line tool that writes in a file the system messages, including the errors and messages generated by the application. Open a terminal and execute the `adb` file:

```
danielbastos@Quobis-danielbastos:~/Logs Android/platform-tools_r31.0.0-linux/platform-tools$ ./adb logcat
* daemon not running; starting now at tcp:5037
* daemon started successfully
- waiting for device -
```

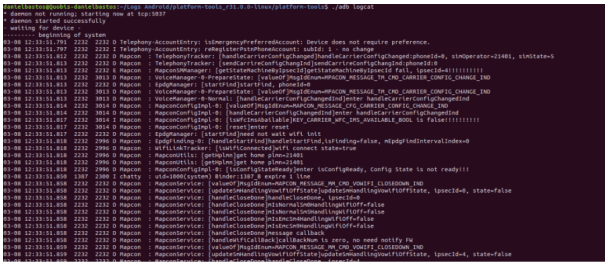
When this is done, connect the Android device to the PC and select “File transfer” in the connection options:




Next, enable the developer options in the mobile device. This is usually done by tapping several times in the Build Number. Then go to Developer options and enable USB debugging.

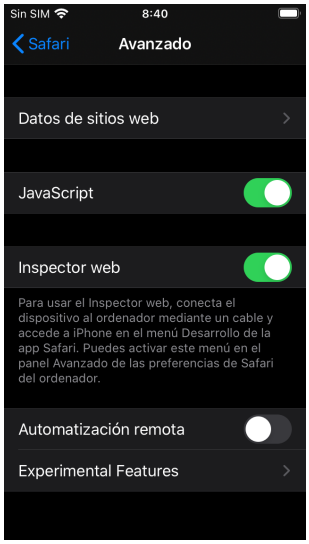
Finally, allow USB debugging for the PC.

Now, you can open the application that you want to diagnose and reproduce the problem. Save the console output from the PC terminal.



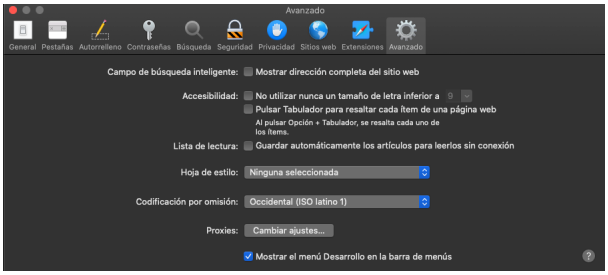
### 12.2.2.2 iOS application

First of all you need to install the debug application on the device. Afterwards, enable the Safari web inspector in Settings > Safari > Advanced > Web Inspector

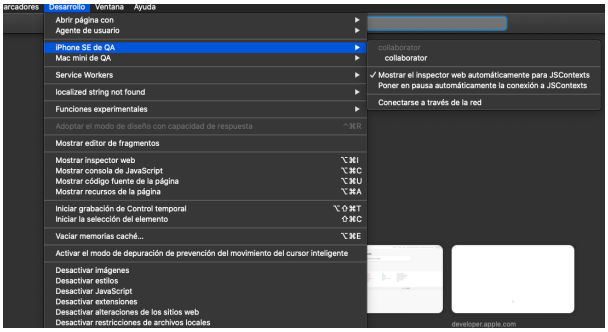



Now connect the iOS device to macOS via wire and enable the developer menu in the macOS Safari:

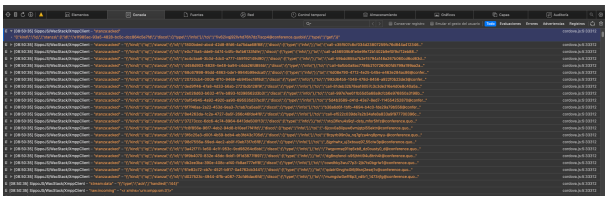
Safari > Preferences > Advanced > Show development menu in the menu bar




Once applied, restart Safari and open the iOS application. If now you open Safari in the macOS and go to Development, you should see the iOS device and the name of the application within.




If you click on the name of the application (Quobis collaborator in the previous example), you will be able to see all the logs that come out through the console.



CHAPTER

THIRTEEN

# QUOBIS WAC REST APIS

The Quobis wac REST API documentation is available in OpenAPI format in the following link:

- Browser-based: Quobis wac REST API reference

**What is the Quobis REST API**

A RESTful API (in short, a REST API) is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data objects used internally by an application. The Quobis wac provides a REST API that enables developers to interact with the backend from external applications. The Quobis REST API allows OSS/BSS integration and different configuration and operation tasks such as user and contact management, presence information and CDR retrieval.

**What is it used for**

The REST API serves as a complement to the client-side SDKs and it is mentioned as "Service API" or "SAPI" in the documentation. It is a key component of the solution as it allows the integration and customization with existing network elements such as customer databases, CRS, policy systems, authentication platforms, directories, etc... ). That means that the platform can adapt its behaviour according to different customer requirements and thus providing a number of different use cases with the same backend.

The administration of the Quobis wac is based on the REST API, all commands to handle the system must be done based on a REST API. You can use CURL or any other tool that allows you to consume the REST API (SAPI). We present here two alternatives:

- **CURL**: to use it on bash commands via the command line interface
- **Postman**: Postman is a popular API client that makes it easy for developers to create, share, test and document APIs via a graphical interface.

**REST API authentication**

All REST API requests must be authenticated using a Authorization header. This is explained in the *Authentication* section.

CHAPTER

# FOURTEEN

# DOCUMENTATION IN PDF

- Quobis wac administration manual