Collabora Online

SDK Manual

by Collabora

Mar 08, 2024

Why integrate

By integrating Collabora Online into your services, you provide your users with excellent solution for document editing and collaboration capabilities, while being gentle on your hardware resources and budget.

High availability	Providing redundancy against fault
Easy to set up	A clean and attractive architecture, easy scalable Server of your choice no local package or software needed perfect to combine with local office applications
Saves money	No need for extra high availability databases, message bus, extra shared storage
Transparent pricing	Unlike other products, no hidden extra support costs
Best document supports	Supports more document formats than any other product, including many legacy formats
Document freedom	Keep the documents file formats, all supported by the global community of interoperability experts
Full open source	Full open source code; no additional features behind pay walls
Best document security	Guarantee, combined with a 'secure view, that users/customers documents never leave the server
Best user experience	Most features and WYSIWYG as on desktop
Partner advantages	Attractive and flexible partner program
Direct support	Support via partner portal, ticketing system and direct help
Development roadmap	Partners have an active voice in the development roadmap

Chapter 2

Installation Guide

2.1 Installation from packages

Collabora deliver signed binary packages for 64-bit Linux distributions.

Starting with version 23.05, the deb and rpm packages can be run on any modern Linux distribution.

Collabora Online Development Edition (CODE) packages available in `x86-64`, `arm64`, and `ppc64le` platforms.

Collabora Online (COOL) packages are made for `x86-64` platform.

2.1.1 The Installation Procedure

On all the supported platforms, the installation procedure consist of three steps:

- Import of the signing key
- The installation itself
- Starting of the service, and enabling it for auto-start after reboot

Collabora's Partner

If you are Collabora's Partner, please log into The Partner portal get your unique secret URL from the Partner portal and follow the instructions listed there.

CODE

If you are **not** Collabora's Partner please follow CODE instructions

Distro-specific Installation Instructions

To install Collabora Office you need system administrator (root) privileges. The following command line examples are supposed to be entered from a system administrator (root) account. Alternatively you can use sudo.

```
export customer_hash=Example-413539ece39485afc35b4a469adfde0a279d2fd2
```

Debian, Ubuntu, other deb based Linux distribution

Please type the following commands into the shell as root:

1. download the signing key

```
cd /usr/share/keyrings
wget
https://collaboraoffice.com/downloads/gpg/collaboraonline-release-keyring.gpg
```

2. add the repository to /etc/apt/sources.list.d

```
cat << EOF > /etc/apt/sources.list.d/collaboraonline.sources
Types: deb
URIs:
https://www.collaboraoffice.com/repos/CollaboraOnline/23.05/customer-deb-$customer_hash
Suites: ./
Signed-By: /usr/share/keyrings/collaboraonline-release-keyring.gpg
EOF
```

3. perform the installation

```
apt update && apt install coolwsd collabora-online-brand
```

After successful installation, please follow the chapter Configuration.

RHEL, CentOS, and their derivatives

Please type the following commands into the shell as root:

1. import the signing key

```
wget
https://collaboraoffice.com/repos/CollaboraOnline/23.05/customer-rpm-$customer_hash/rep
 && rpm --import repomd.xml.key
```

2. add the repository URL to yum

```
yum-config-manager --add-repo
https://collaboraoffice.com/repos/CollaboraOnline/23.05/customer-rpm-$customer_hash
```

3. perform the installation

```
yum install coolwsd collabora-online-brand
```

After successful installation, please follow the chapter Configuration.

SLES 15 / openSUSE Leap 15.x

Please type the following commands into the shell as root:

1. import the signing key

```
wget
https://collaboraoffice.com/repos/CollaboraOnline/23.05/customer-rpm-$customer_hash/rep
 && rpm --import repomd.xml.key
```

2. add the repository URL to zypper

```
zypper ar -t yum
"https://collaboraoffice.com/repos/CollaboraOnline/23.05/customer-rpm-$customer_hash"
"Collabora Online"
```

3. perform the installation

```
zypper ref && zypper in coolwsd collabora-online-brand
```

After successful installation, please follow the chapter Configuration.

How to upgrade

If you are upgrading from Collabora Online 6.4 or earlier version to version 21.11 or newer, please read Upgrade to Collabora Online 21.11 Otherwise it is enough to change the version number in the repository URL and upgrade as usual with the respective package manager. Although upgrade

process is safe, it is always a good idea to backup configuration files in `/etc/coolwsd/` just in case.

Localization

For complete user interface localization you need to install Collabora Office language resources. They are not direct dependencies of coolwsd. For example for German dialogs on Debian/Ubuntu:

```
apt install collaboraoffice*de
```

Spelling dictionaries and thesauri

Collabora Online can use internal spelling dictionaries and thesauri (`collaboraoffice*-dict-*` packages). Collabora Online can use system spelling dictionaries and thesauri, too, that are located in `/usr/share/hunspell` and `/usr/share/mythes` directories.

Additionally, and starting with version 22.05, it's possible to enable support for external grammar checking using LanguageTool. For information please consult Language Tool.

2.2 Docker images

As an alternative to native packages, Collabora Productivity provide pre-built Collabora Online Docker images, as well as scripts and Dockerfile's to create a Collabora Online Docker images.

You either need native packages, or a Docker image, not both.

2.2.1 Pre-made Docker images

The CODE Docker image can be installed to any `x86-64`, `ppc64le` or `arm64` host, and it is fully configurable. For more information about setup and configuration for deployment, please read the CODE Docker page. If you want to try it out quickly, you can set up CODE docker image with file sharing integration in less than 5 minutes in a very basic way, following these instructions: quick tryout with ownCloud or quick tryout with Nextcloud.

For customers Collabora provide pre-built Collabora Online images for the x86-64 platform. - `registry.gitlab.collabora.com/collabora-online/docker` is the publicly available license key enabled version. License keys can be purchased via partners or from Collabora directly. - `registry.gitlab.collabora.com/productivity/collabora-online` is the full version of Collabora online for customers. If you are a customer, and need this image, please contact support.

Docker images are tagged with the version number, however, usually it is the best to use the `:latest`.

2.2.2 Build Docker image

Docker images can be created on demand from the latest version of Collabora Online and the underlying system components. Please find everything in Collabora Online source code repository on GitHub. Docker images can be built from packages or from source code. The provided Dockerfile is a working sample. Feel free to add more packages to it, for example more fonts, if you need them.

2.2.3 Create a container from the image and run it

For more information please refer to the CODE Docker page, configuration options are the same for all containers.

2.3 CODE Docker image

The CODE Docker image can be installed to any x86-64 or arm64 host, and it is fully configurable.

2.3.1 How to grab the CODE image from Docker image

Collabora Online Development Edition (CODE) is available as a Docker image from Docker Hub. Currently, the supported platforms are `x86-64`, `ppc64le` and `arm64`, and the image was mostly tested on Linux. If you are not familiar with Docker concepts and basic commands, read the Docker Get Started document first.

Listing 2.1. Grab the Docker image

```
docker pull collabora/code
```

Listing 2.2. Start a new container:

```
docker run -t -d -p 127.0.0.1:9980:9980 collabora/code
```

This is the minimal command line to start a new container. There are a few optional and recommended command line options:

- `--name collabora` gives a specific name to the container instead of a random one.
- `--restart always` restarts the container after a crash that may occur.
- `--privileged` starts the container with rights required for faster jail creation via bind mount.
- `-p '[::1]:9980:9980'` adds a port redirection for IPv6. If your host has IPv6 configured, you may want to add this if you get "Connection Refused" errors.
- with `-e` you can pass environment variables to the container, see below.

2.3.2 How to configure Docker image

There are multiple ways to put application configuration into Docker containers. Collabora Online has many configuration options and the Docker image comes with a built-in `/etc/coolwsd/coolwsd.xml` configuration file with the defaults.

1. Setting the application configuration dynamically via environment variables

After the `-e` command line option of `docker run` command you can define environment variables, that are passed to the container.

By default Collabora Online enables the first WOPI host that tries to connect. You can define the allowed WOPI hosts by passing environment variables.

`aliasgroup1=https://<domain1>:443,https://<your-dot-escaped-aliasname1>|<your-dot-escap`
`aliasgroup1, aliasgroup2...` and so on should be added as per the requirement. `<domain1>` is the WOPI host, i.e. your preferred File Sync and Share solution that implements the WOPI protocol, for example `share.example.com`. `<your-dot-escaped-aliasname1>|<your-dot-escaped-aliasname2>` are the aliasnames with which you can access the same WOPI host (in this case `<domain1>`) aliasnames can use regular expressions.If you don't have any aliases, then only domain needs to be defined, for example `aliasgroup2=https://<domain2>:443`.

Other optional environment variables that you can pass to the container at startup are the following:

`username`	User name for the Section 2.7.10
`password`	Password for the Section 2.7.10
`DONT_GEN_SSL_CERT`	When this environment variable is set (is not ""), then startup script will not generate a new SSL certificate signed by a dummy CA. It is useful, if you want to use your own SSL certificate for some reason.
`cert_domain`	When this environment variable is set (is not ""), then startup script will generate a new SSL certificate signed by a dummy CA for this domain, not for `localhost`

`server_name`	When this environment variable is set (is not ""), then its value will be used as server name in `/etc/coolwsd/coolwsd.xml`. Without this, CODE is not delivering a correct host for the websocket connection in case of a proxy in front of it.
`dictionaries`	By default only limited set of spelling dictionaries and thesauri are configured for CODE, mainly for performance reasons. The default set of languages is the following: `de_DE en_GB en_US es_ES fr_FR it nl pt_BR pt_PT ru`. With the dictionaries environment variable you can change this list. The dictionaries environment variable should contain the space separated list of language codes (optionally followed by country code). In order to save resources, it makes sense to load only those dictionaries that are actually needed.
`extra_params`	You can pass extra command line parameters to `coolwsd` via this environment variable. For example, if you want to start coolwsd without SSL, when you test or develop, the syntax is: `-e "extra_params=--o:ssl.enable=false"`. To learn about all possible options, refer to the self-documented `/etc/coolwsd/coolwsd.xml` configuration file in the Docker image.

2. Use the configuration file directly

After starting the container, you can copy the configuration file out of the container (using `docker cp`), edit it, and copy it back to the container. It is also possible to mount the configuration file, and modify it outside of the container. The container will notice that the configuration file has changed, and the service will be restarted (don't forget the `--restart always` option when you start the container with `docker run`).

Troubleshooting

After starting of the container, try:

```
curl -k https://localhost:9980
```

You should get the OK string, if everything is in order. Otherwise, you can check the log with:

```
docker logs collabora
```

Misc

If you need customizations, for example additional fonts, you can build the docker image yourself. Please find everything in Collabora Online source code repository on GitHub.

2.4 Collabora Online for Kubernetes

In order for Collaborative Editing and copy/paste to function correctly on kubernetes, it is vital to ensure that all users editing the same document and all the clipboard request end up being served by the same pod. Using the WOPI protocol, the https URL includes a unique identifier (WOPISrc) for use with this document. Thus load balancing can be done by using WOPISrc – ensuring that all URLs that contain the same WOPISrc are sent to the same pod.

2.4.1 Deploying Collabora Online in Kubernetes

1. Install helm
2. Setting up Kubernetes Ingress Controller
    1. Nginx:

   Install Nginx Ingress Controller

2. HAProxy:

Install HAProxy Ingress Controller

Note Openshift uses minimized version of HAproxy called Router that doesn't support all functionality of HAProxy but for COOL we need advance annotations Therefore it is recommended deploy HAproxy Kubernetes Ingress in `collabora` namespace

---

3. Create an `my_values.yaml` (if your setup differs e.g. take an look in then `values.yaml ./collabora-online/values.yaml`) of the helmchart

1. HAproxy:

```yaml
replicaCount: 3

ingress:
   enabled: true
   className: "haproxy"
   annotations:
      haproxy.org/timeout-tunnel: "3600s"
      haproxy.org/backend-config-snippet: |
         balance url_param WOPISrc check_post
         hash-type consistent
   hosts:
      - host: chart-example.local
        paths:
        - path: /
          pathType: ImplementationSpecific

image:
   tag: "latest"

autoscaling:
   enabled: false

collabora:
   aliasgroups:
      - host: "https://example.integrator.com:443"
   extra_params: --o:ssl.enable=false --o:ssl.termination=true

resources:
   limits:
      cpu: "1800m"
      memory: "2000Mi"
   requests:
      cpu: "1800m"
      memory: "2000Mi"
```

2. Nginx:

```yaml
replicaCount: 3

ingress:
   enabled: true
   className: "nginx"
   annotations:
      nginx.ingress.kubernetes.io/upstream-hash-by: "$arg_WOPISrc"
      nginx.ingress.kubernetes.io/proxy-body-size: "0"
      nginx.ingress.kubernetes.io/proxy-read-timeout: "600"
      nginx.ingress.kubernetes.io/proxy-send-timeout: "600"
   hosts:
```

```
      - host: chart-example.local
        paths:
        - path: /
          pathType: ImplementationSpecific

image:
   tag: "latest"

autoscaling:
   enabled: false

collabora:
   aliasgroups:
      - host: "https://example.integrator.com:443"
   extra_params: --o:ssl.enable=false --o:ssl.termination=true

resources:
   limits:
      cpu: "1800m"
      memory: "2000Mi"
   requests:
      cpu: "1800m"
      memory: "2000Mi"
```

Note

- **Horizontal Pod Autoscaling(HPA) is disabled for now. Because after scaling it breaks the collaborative editing and copy/paste. Therefore please set replicaCount as per your needs**
- If you have multiple host and aliases setup set aliasgroups in `my_values.yaml`:

```
collabora:
   - host: "<protocol>://<host-name>:<port>"
     # if there are no aliases you can ignore the below line
     aliases: ["<protocol>://<its-first-alias>:<port>,
<protocol>://<its-second-alias>:<port>"]
   # more host and aliases list is possible
```

- Specify `server_name` when the hostname is not reachable directly for example behind reverse-proxy

```
collabora:
   server_name: <hostname>:<port>
```

- In **Openshift** , it is recommended to use HAproxy deployment instead of default router. And add `className` in ingress block so that Openshift uses HAProxy Ingress Controller instead of `Router`:

```
ingress:
   className: "haproxy"
```

4. Install helm-chart using below command, it should deploy the collabora-online

```
helm repo add collabora https://collaboraonline.github.io/online/
helm install --create-namespace --namespace collabora collabora-online
collabora/collabora-online -f my_values.yaml
```

5. Follow only if you are using `NodePort` service type in HAProxy and/or using minikube to

setup, otherwise skip

1. Each container port is mapped to a `NodePort` port via the `Service` object. To find those ports

```
kubectl get svc --namespace=haproxy-controller
```

Example output:

```
|----------------|---------|-------------|------------|-----------------------------
|NAME            |TYPE     |CLUSTER-IP    |EXTERNAL-IP |PORT(S)
                              |
|----------------|---------|-------------|------------|-----------------------------
|haproxy-ingress |NodePort |10.108.214.98 |<none>
|80:30536/TCP,443:31821/TCP,1024:30480/TCP |
|----------------|---------|-------------|------------|-----------------------------
```

In this instance, the following ports were mapped:

- Container port `80` to `NodePort 30536`
- Container port `443` to `NodePort 31821`
- Container port `1024` to `NodePort 30480`

6. Additional step if deploying on minikube for testing:

1. Get minikube ip:

```
minikube ip
```

Example output:

```
192.168.0.106
```

2. Add hostname to `/etc/hosts`

```
192.168.0.106   chart-example.local
```

3. To check if everything is setup correctly you can run:

```
curl -I -H 'Host: chart-example.local' 'http://192.168.0.106:30536/'
```

It should return a similar output as below:

```
HTTP/1.1 200 OK
last-modified: Tue, 18 May 2021 10:46:29
user-agent: COOLWSD WOPI Agent 6.4.8
content-length: 2
content-type: text/plain
```

2.4.2 Kubernetes cluster monitoring

1. Install kube-prometheus-stack, a collection of Grafana dashboards, and Prometheus rules combined with documentation and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with Prometheus using the Prometheus Operator.
2. Enable prometheus service monitor, rules and grafana in your `my_values.yaml`

```
prometheus:
   servicemonitor:
      enabled: true
      labels:
         release: "kube-prometheus-stack"
```

```
   rules:
      enabled: true # will deploy alert rules
      additionalLabels:
         release: "kube-prometheus-stack"
grafana:
   dashboards:
      enabled: true # will deploy default dashboards
```

Note Use `kube-prometheus-stack` as release name when installing kube-prometheus-stack helm chart because we have passed `release=kube-prometheus-stack` label in our `my_values.yaml`. For Grafana Dashboards you may need to enable scan in correct namespaces (or ALL), enabled by `sidecar.dashboards.searchNamespace` in Helmchart of grafana (which is part of PrometheusOperator, so `grafana.sidecar.dashboards.searchNamespace`)

2.4.3 Dynamic/Remote configuration in kubernetes

For big setups, you may not want to restart every pod to modify WOPI hosts, therefore it is possible to setup an additional webserver to serve a ConfigMap for using Remote/Dynamic Configuration

```
collabora:
   env:
      - name: remoteconfigurl
        value: https://dynconfig.public.example.com/config/config.json

 dynamicConfig:
   enabled: true

   ingress:
      enabled: true
      annotations:
      "cert-manager.io/issuer": letsencrypt-zprod
      hosts:
      - host: "dynconfig.public.example.com"
      tls:
      - secretName: "collabora-online-dynconfig-tls"
         hosts:
            - "dynconfig.public.example.com"

   configuration:
      kind: "configuration"
      storage:
         wopi:
         alias_groups:
            groups:
            - host: "https://domain1\\.xyz\\.abc\\.com/"
               allow: true
            - host: "https://domain2\\.pqr\\.def\\.com/"
               allow: true
               aliases:
                  - "https://domain2\\.ghi\\.leno\\.de/"
```

Note In current state of COOL `remoteconfigurl` for Remote/Dynamic Configuration should be HTTPS.

2.4.4 Useful commands to check what is happening

Where is this pods, are they ready?

```
kubectl -n collabora get pod
```

example output :

```
NAME                                  READY   STATUS    RESTARTS   AGE
collabora-online-5fb4869564-dnzmk   1/1     Running   0          28h
collabora-online-5fb4869564-fb4cf   1/1     Running   0          28h
collabora-online-5fb4869564-wbrv2   1/1     Running   0          28h
```

What is the outside host that multiple coolwsd servers actually answering?

```
kubectl get ingress -n collabora
```

example output :

```
|-----------|------------------|--------------------------|------------------------|-------|
| NAMESPACE |       NAME       |           HOSTS          |         ADDRESS        |
 | PORTS |
|-----------|------------------|--------------------------|------------------------|-------|
| collabora | collabora-online |chart-example.local       |
 |  80   |
|-----------|------------------|--------------------------|------------------------|-------|
```

To uninstall the helm chart

```
helm uninstall collabora-online -n collabora
```

2.5 Fonts

Collabora Online uses Collabora Office as its backend, which comes with a large variety of free fonts, see the list below:

- `Caladea` and `Carlito`, which are metric-compatible with `Cambria` and `Calibri`
- `Déja Vu`
- `Emoji One`
- `Gentium`
- `Google Open Sans` and `PT Serif`
- `Google Noto` (full Unicode coverage)
- `Karla`
- `Liberation Sans` and `Liberation Serif`, which are metric-compatible with `Arial` and `Times New Roman`
- `Linux Libertine G`
- `Source Code Pro` and `Source Sans Pro`

When you install coolwsd package, the post-install script will look for additional fonts on your system, and install them for Collabora Online (in the systemplate). If you install fonts to your system after installing coolwsd, you need to update the systemplate manually and restart coolwsd service.

In Collabora Online 22.05.7 the possibility of remote font downloading was introduced. This method does not require restart of the coolwsd service, remote fonts become available for new editing sessions within a minute. See the details in Remote configuration chapter.

2.6 Updating systemplate

Each document is isolated in its own `chroot` jail running its own instance of a LibreOfficeKit process, and runs as a non-privileged "cool" user. These `chroot` jails contain only the bare minimum of files (libraries, fonts, etc.) needed for running Collabora Office (LibreOfficeKit). The template of the jails is called *systemplate*, it is located at `/opt/cool/systemplate`, and it is generated after installation of the coolwsd package. The systemplate is also re-generated after installing updates of packages that are in use in systemplate (on RPM based systems) or after a successful apt update (on DEB based systems).

However, it is possible that the user wants to build systemplate manually, for example when new fonts are installed, or a security update of system libraries is deployed by other means. Perform the following command as root user.

In Collabora Online 21.11 and newer:

```
coolconfig update-system-template
```

In Collabora Online 6.4 and older:

```
loolconfig update-system-template
```

2.7 Configuration

The postinstall script of coolwsd package added a non-privileged user to the system: *cool* . Collabora Online service will be run by *cool* user. Also the service was registered to systemd, enabled on system start and started. Useful commands:

- `systemctl enable coolwsd` – enable coolwsd on system start
- `systemctl disable coolwsd` – disable coolwsd on system start
- `systemctl status coolwsd` – check status of coolwsd
- `systemctl stop coolwsd` – stop coolwsd service
- `systemctl start coolwsd` – start coolwsd service
- `systemctl restart coolwsd` – stop then start coolwsd service
- `journalctl -u coolwsd` – read the log produced by coolwsd

Collabora Online has to be configured before use. Most of the options have sensible defaults.

Collabora online has layered configuration, which means that settings are read from `/etc/coolwsd/coolwsd.xml` but can be overridden by command line switches (for example in systemd's coolwsd.service file). By using `--o:name=value` the setting called `name` can be replaced by `value`. For example: `--o:per_document.max_concurrency=12`. This will override the `max_concurrency` to 12, regardless of what the XML has set.

Default configuration entries and values are set before loading the configuration file from disk. This ensures that an upgrade to the server with new configuration entries will not break the server when the XML is not upgraded, rather, the server will fallback to the defaults when it fails to find the entry in the XML.

The coolwsd service has to be restarted after a change in configuration.

2.7.1 User interface settings

With Collabora Online 6.4 the systems administrator can set the `classic` menu + toolbar user interface or the new `notebookbar` user interface. See the `user_interface.mode` setting in the configu-

ration file.

With Collabora Online 21.11 the use of `classic` and `notebookbar` is deprecated , use `compact` for `classic` and use `tabbed` for `notebookbar`. See the `user_interface.mode` setting in the configuration file.

2.7.2 Network settings

Collabora Online can use IPv4, IPv6 or both. By default it uses both. See the `net.proto` setting config file.

It is possible for a coolwsd server to bind to localhost only, which makes sense, when it is used behind a reverse proxy. The corresponding setting is `net.listen`.

It is possible to use a different service root than the toplevel. If the rules of your organization do not permit running services in the root, you can use a subpath for it, like `https://example.org/IT/CollaboraOnline` by setting `/IT/CollaboraOnline` as the `net.service_root` in the configuration file.

2.7.3 SSL configuration

Collabora Online uses WOPI protocol, which mandates SSL. However, it is possible to run Collabora Online server without SSL, it is configurable. Basically there are 3 modes:

1. SSL
2. SSL termination
3. No SSL

When SSL is enabled, in `/etc/coolwsd/coolwsd.xml` the path to SSL key, SSL certificate and SSL CA certificate has to be given in the ssl block. This also implies that it is recommended to run coolwsd from a server which name is in DNS (e.g. hostname.example.com), and it has proper SSL certificate. Restart coolwsd, check the status of the service, and if it is running, you can try if you can connect to it via SSL:

```
curl -v https://hostname.example.com:9980/hosting/discovery
```

If it fails, you have to debug SSL settings.

For testing purposes it is OK to use self signed certificates. Since Collabora Online 2.1 we no longer ship self signed certificate for localhost, for security reasons. You can create the necessary files yourself. The following example creates a certificate for hostname.example.com by a newly created dummy certificate authority. The resulting `.pem` files are copied to default configuration directory of coolwsd.

```
 mkdir -p /opt/ssl/
 cd /opt/ssl/
 mkdir -p certs/ca
 openssl genrsa -out certs/ca/root.key.pem 2048
 openssl req -x509 -new -nodes -key certs/ca/root.key.pem -days 9131 -out
certs/ca/root.crt.pem -subj "/C=DE/ST=BW/L=Stuttgart/O=Dummy Authority/CN=Dummy
Authority"
 mkdir -p certs/{servers,tmp}
 mkdir -p "certs/servers/hostname.example.com"
 openssl genrsa -out "certs/servers/hostname.example.com/privkey.pem" 2048 -key
"certs/servers/hostname.example.com/privkey.pem"
 openssl req -key "certs/servers/hostname.example.com/privkey.pem" -new -sha256
-out "certs/tmp/hostname.example.com.csr.pem" -subj
"/C=DE/ST=BW/L=Stuttgart/O=Dummy Authority/CN=hostname.example.com"
 openssl x509 -req -in certs/tmp/hostname.example.com.csr.pem -CA
certs/ca/root.crt.pem -CAkey certs/ca/root.key.pem -CAcreateserial -out
certs/servers/hostname.example.com/cert.pem -days 9131
 mv certs/servers/hostname.example.com/privkey.pem /etc/coolwsd/key.pem
```

```
mv certs/servers/hostname.example.com/cert.pem /etc/coolwsd/cert.pem
mv certs/ca/root.crt.pem /etc/coolwsd/ca-chain.cert.pem
```

The SSL termination option in the config file enables integration of Collabora Online with SSL termination proxies, which handle incoming SSL connections, decrypt the SSL and pass on the unencrypted request to the server. In this setup only the proxy server has to have proper SSL settings, Collabora Online server is hidden behind it, and Collabora Online communicates unencrypted with the proxy.

If you set both `enable` and `termination` settings to `false` in `/etc/coolwsd/coolwsd.xml`, then Collabora Online can be used in a HTTP-only environment, without encryption between browser and server. It is not recommended to use Collabora Online in this mode, but for testing only it is OK.

You can set the list of accepted SSL ciphers with the `cipher_list` setting. The default cipher list is: `ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH`.

2.7.4 Security settings

Security settings are configurable, and coolwsd is allowed to run without seccomp and capabilities. There are some significant security trade-offs here which are now at least configurable. It is recommended to use the defaults. See the security section in `/etc/coolwsd/coolwsd.xml`.

2.7.5 Validating digital signatures

Collabora Online uses NSS (Mozilla's Network Security Services) for validation of digital signatures. NSS comes with default configuration that includes a few trusted root CAs, but users may want to import their own trusted root CAs. The `certificates.database_path` configuration option in `/etc/coolwsd/coolwsd.xml` specifies the NSS certificate database that should be used with Collabora Online. This database should be readable by the coolwsd process. Custom root certificates can be imported into this database. For detailed instructions about creating NSS certificate database and importing certificates, please refer to the manual of the certutil tool, that is provided by mozilla-nss-tools, nss-tools or libnss3-tools package, depending on the Linux distribution.

2.7.6 Backend storage configurations

Currently there are two backend storages are implemented: **file system** and **WOPI**.

File system storage is disabled by default, and should not be used in production environment. It is insecure by nature, because it serves any file that the *cool* user can read from the local file system, including `/etc/coolwsd/coolwsd.xml`, `/etc/passwd` and so on. It can be used for testing only. To enable:

Listing 2.3. in storage block of coolwsd.xml

```
<filesystem allow="true" />
```

or

Listing 2.4. in command line

```
--o:storage.filesystem[@allow]=true
```

WOPI on the other hand is the recommended backend storage. WOPI is Web Application Open Platform Interface, a protocol based on open standard for remote document access with authentication. Collabora Online accepts connection requests only from trusted WOPI hosts. The administrator has to list the host names and/or IP addresses of these trusted WOPI hosts in the storage.wopi block. Please note that connection requests from the same machine are always accepted.

2.7.7 Logging

See the `<logging>` section in `/etc/coolwsd/coolwsd.xml`. Set the log level and verbosity to one of: none (turns off logging), fatal, critical, error, warning, notice, information, debug, trace. The default log level is warning. If `<color>` is set to true, then *coolwsd* will generate logging information

containing console color codes. It is possible to redirect logs to a file. The trace file defined in `<trace>` section provides extra debug information.

2.7.8 Performance

There are two performance related settings.

One is `num_prespawn_children`. It is the number of child processes to keep started in advance and waiting for new clients. More prespawn children consume more memory, but server answers more quickly to requests under load. The default is `1`.

The other is `per_document.max_concurrency` which limits the number of threads to use while processing a document. The default here is `4`.

2.7.9 Allowed dictionary languages

When there are a lot of spellchecker dictionaries and thesauri installed on a system, it may take considerable time at startup to preload them. Therefore by default, only dictionaries for the following languages are enabled:

Listing 2.5. list is controlled by the `allowed_languages` setting, you can add or remove language tags as needed.

```
de_DE en_GB en_US es_ES fr_FR it pt_BR pt_PT ru
```

2.7.10 Admin Console

You can do live monitoring of all the user sessions running on Collabora Online instance. The Admin Console URL is:

```
https://*hostname*:*port*/browser/dist/admin/admin.html
```

Port is `9980` by default. It will ask for username and password which is set in the admin_console block of `/etc/coolwsd/coolwsd.xml` or by `--o:admin_console.username=username` and `--o:admin_console.password=password` in coolwsd command line. You must set username and password. Admin Console is disabled if either of these are not set.

Note: it is possible to set up a password that is stored as salted hash in the config file, instead of plain text. This is the recommended way to set up password for the Admin Console. Use the `coolconfig` utility.

Note: there is support for authentication with `PAM`, if it is set up for coolwsd in the system. For example, with a simple `/etc/pam.d/coolwsd` config below, the user which runs coolwsd ('cool' in production environment) can login to admin console with normal linux password.

```
auth required pam_unix.so
account required pam_unix.so
```

After entering the correct password you should be able to monitor the live documents opened, total users, memory consumption, document URLs with number of users viewing that document etc. You can also kill the documents directly from the panel which would result in closing the socket connection to the respective document.

The admin-console front-end presents and fetches its data via a defined web socket protocol, which can be used to collect information programatically to integrate with other monitoring and control solutions. For the websocket protocol details of Admin Console, see the Admin Console section in the protocol documentation:

https://github.com/CollaboraOnline/online/blob/master/browser/README

and

https://github.com/CollaboraOnline/online/blob/master/wsd/protocol.txt .

It is simple to subscribe to receive client notifications, query the open documents and change server

settings.

2.7.11 Monitoring usage metrics

Collabora Online is capable of providing a wide variety of metrics related to system usage, eg. memory or CPU use, number of running kit processes or views in document sessions etc. The metrics can be retrieved in Prometheus-compatible format via the following URL:

```
https://*hostname*:*port*/cool/getMetrics
```

The URL uses the same authentication as the admin console. The available metrics are listed at: https://github.com/CollaboraOnline/online/blob/master/wsd/metrics.txt .

2.7.12 Feature Locking

Collabora Online provides a way to lock the user out from using certain features or make them read-only users. When a user clicks on a feature that is locked, the user will be prompted with a popup with details about unlocking. To disable any feature you can specify its UNO command in the `locked_commands` field. When a locked user is treated with read-only permission `locked_commands` option is ignored. Other related options can be found in coolwsd.xml. To mark a user locked, the WOPI client should return `CheckFileInfo` containing a field `IsUserLocked` with a boolean value. To make the locked users read-only set `is_lock_readonly` setting to true.

To allow/deny feature_lock per WOPI host:

```
<locked_hosts desc="Allow/deny feature-locked hosts. When allowed, the below host
specification overrides the CheckFileInfo response." allow="true">
  <fallback desc="What to do if the given host is not covered by any rule in
locked_hosts" read_only="true" disabled_commands="true"/>
  <host desc="Regex pattern of hostname to set as full-featured or locked."
read_only="false" disabled_commands="false">pattern1</host>
</locked_hosts>
```

Please note that locked_hosts allow should be true to enable allow/deny feature_lock per WOPI host. If host pattern does not match for locked_host the fallback setting will be applied. This also overwrites the `isUserLocked` CheckFileInfo response.

Explaining all cases possible with above settings:
case-1: CheckFileInfo property `isUserLocked` is passed and `locked_hosts`'s `allow=true` then COOL will follow the `locked_hosts` settings i.e. if host is mentioned in `locked_hosts` then it will follow that settings else it will follow `fallback` settings
case-2: If `is_lock_readonly` is set to `true` COOL will make all the `locked_host` readonly who has `disabled_commands` set to true

2.7.13 Feature Restriction

Collabora provides a way to completely disable/hide certain features from the user. You can specify the feature's UNO command to disable it in `restricted_commands`. To mark a user restricted, the WOPI client should return `CheckFileInfo` containing a field `IsUserRestricted` with a boolean value.

2.7.14 Multihost Configuration

To use multiple host and aliases with one COOL server you have to set alias_groups mode attribute to 'groups' and define group for each intergrator's instance.

For example:
* If you have two hosts:
    1. host1 has no aliases
    2. host2 has aliases like aliasname1, aliasname2, aliasname3. For all these aliases regex would be aliasname[0-9]{1}.

```
<alias_groups desc="default mode is 'first' it allows only the first host when
groups are not defined. set mode to 'groups' and define group to allow multiple
host and its aliases" mode="groups">
<!-- If you need to use multiple wopi hosts, please change the mode to "groups"
and
add the hosts below.  If one host is accessible under multiple ip addresses
or names, add them as aliases. -->

   <group>
      <host desc="hostname to allow or deny."
allow="true">https://host1:443</host>
   </group>

   <group>
      <host desc="hostname to allow or deny."
allow="true">https://host2:443</host>
      <alias desc="regex pattern of
aliasname">https://aliasname[0-9]{1}:443</alias>
   </group>

   <!-- More "group"s possible here -->
</alias_groups>
```

You can add multiple groups. Here host1 and host2 can be any service COOL has integration with Available integrations.

2.7.15 Remote/Dynamic Configuration

The following new configuration makes it possible to:
1. Allow or deny WOPI host for single and multihost configuration i.e. alias_groups
2. Make WOPI host read-only or locked i.e. locked_hosts
3. Setting remote_font_config url
4. Customize unlock dialog

These changes can be now done without restarting Collabora server. Collabora server will request a JSON response to the remote server every 60 second and if there is new changes in JSON it will overwrite coolwsd.xml settings. Thus, adding the respective WOPI hosts and locked_hosts to the allow/-deny list. The configuration will take effect the next time a document gets open.

Note: Collabora uses ETag header to identify whether the JSON is changed from last request or not. Therefore it is recommended to add ETag header in JSON response of remote server

Enable remote server configuration by adding url

```
<remote_config >
  <remote_url desc="remote server to which you will send request to get remote
config in response" type="string"
default="">https://server_url_endpoint</remote_url>
</remote_config>
```

JSON format

Configuration will be overwritten if JSON response has been changed

```
{
  "kind": "configuration",
  "remote_font_config":
  {
    "url": "https://.../fonts.json"
  },

  "storage":
  {
```

```
    "wopi":
    {
      "alias_groups":
      {
        "mode" : "groups",
        "groups":
        [
          { "host": "scheme://hostname:port", "allow": "true" , "aliases":
["scheme://aliasname1:port", "scheme://alias-regex-pattern:port"]},
        ]
      }
    }
  },

  "feature_locking":
  {
    "locked_hosts":
    {
      "allow":"true",
      "hosts":
      [
          { "host": "pattern1", "read_only": true, "disabled_commands": true },
          { "host": "pattern2", "read_only": false, "disabled_commands": true },
      ]
    },
    // unlock dialog customization
    "unlock_image": "https://<hostname>/static/<image_endpoint>",
    "translations":
      [
        {
          "language": "de",
          "unlock_title": "Gehen Sie zur Detailseite und entdecken Sie alle
Funktionen:",
          "writer_unlock_highlights": "Überprüfen und schreiben Sie mit
Leichtigkeit",
          "calc_unlock_highlights": "Machen Sie sich ein besseres Bild von Ihren
Daten",
          "impress_unlock_highlights": "Bringen Sie Ihre nächste Präsentation auf
den Punkt",
          "draw_unlock_highlights": "Zeichne und organisiere dich",
        },
        // more translations possible
      ]
  },
}
```

Please note that JSON response is checked every minute for changes. Every block in JSON is optional i.e. you can use any of the `remote_font_config`, `storage`, `feature_locking` JSON individually.

JSON config will overwrite values given in coolwsd.xml file i.e. `groups` tag will overwrite all host tags in wopi section. `locked_hosts` allow should be true to enable allow/deny feature_lock per WOPI host. If host pattern does not match for `locked_host`, the fallback setting will be applied. `unlock_image` and `translations` will overwrite respective xml:value pair in feature_locking section

Enable download and availability of more fonts by pointing to a font configuration file

```
<remote_font_config>
    <url desc="URL of optional JSON file that lists fonts to be included in
Online" type="string" default="">https://someserver/path/file.json</url>
</remote_font_config>
```

Remote font configuration JSON format

The JSON file pointed to by the above should have contents like in this example

```
{
    "kind": "fontconfiguration",
    "server": "Some pretty name",
    "fonts": [
        {
            "uri": "https://somehost/path/f1.ttf"
        },
        {
            "uri": "https://someotherhost/path/f2.ttf",
            "stamp": "foo3"
        },
        {
            "uri": "https://whatever/path/x42.ttf"
        }
    ]
}
```

The JSON file is re-downloaded and scanned whenever it has changed. This is checked once a minute.

If an element in the fonts array has a `stamp` property then the font file will be re-downloaded and taken into use whenever the stamp has been changed. (And the old version of the font is forgotten.) The stamp can be any string, its contents is not interpreted in any way. The only thing checked is whether it changes. If no stamp property is provided, the web server in question is queried once a minute to check whether the font file has been updated.

The name of a font file is irrelevant. The name of the font is read from the contents of the file. The file should be a TrueType or OpenType font.

2.7.16 Other settings

See `/etc/coolwsd/coolwsd.xml` for other settings, everything is documented there.

2.8 Proxy settings

Server part of Collabora Online (coolwsd daemon) is listening on port 9980 by default, and clients should be able to communicate with it through port `9980`. Sometimes it is not possible, for example a corporate firewall can allow only ports of well known services, such as port 80 (HTTP) and port 443 (HTTPS). The coolwsd daemon is configurable. It can use other ports than `9980`. Port can be set by the command line option `--port`. However we cannot use for example port `443`, when a web server is running on the same server, which is already bound to port `443`. Reverse proxy setup is also required if you would like to setup load balancing.

> **Attention!** In some cases CODE can be packaged as part of integrator's image and so, it might have different set of instructions. Thus, the easiest way to configure reverse proxy might be better documented in the integrator's documentation.

2.8.1 Reverse proxy with Apache 2 webserver

We assume that coolwsd and Apache2 are running on the same server: `collaboraonline.example.com`. For this to work, you have to follow the steps below:

- Set the server name in Collabora Online configuration
- Enable the required Apache2 modules
- Add reverse proxy settings to Apache2 configuration file

Configure Collabora Online

Collabora Online's configuration file is `/etc/coolwsd/coolwsd.xml`.

The proxy redirects incoming requests to `127.0.0.1`, but replies from coolwsd server must contain the original host name, otherwise the connection will fail. The service can usually figure out the external host name, except in more complex cases. In that case look for the setting `server_name` (empty by default), and enter the host name here, for example `collaboraonline.example.com`.

Required Apache2 modules

Apache2 web server is modular. We need to enable the required modules for this reverse proxy setup. We can use the `a2enmod` command to enable modules. If a module has been enabled already, nothing happens.

- Enable proxy in general: `a2enmod proxy`
- Enable proxy for HTTP protocol: `a2enmod proxy_http`
- Enable SSL support: `a2enmod proxy_connect`
- Enable proxy of websockets: `a2enmod proxy_wstunnel`

On CentOS / RHEL there is no `a2enmod` available. Enabling the modules has to be done by adjusting a config file and adding the LoadModule manually. See server-world.info on CentOS.

Reverse proxy settings in Apache2 config (SSL)

These lines should be inserted into `<VirtualHost>` definition of the site.

In `coolwsd.xml` the corresponding setting is `ssl.enable=true`.

```
 ########################################

 # Reverse proxy for Collabora Online   #

 ########################################


 AllowEncodedSlashes NoDecode
 SSLProxyEngine On
 ProxyPreserveHost On


 # cert is issued for collaboraonline.example.com and we proxy to localhost
 SSLProxyVerify None
 SSLProxyCheckPeerCN Off
 SSLProxyCheckPeerName Off


 # static html, js, images, etc. served from coolwsd
 # browser is the client part of Collabora Online
 ProxyPass           /browser https://127.0.0.1:9980/browser retry=0
 ProxyPassReverse    /browser https://127.0.0.1:9980/browser


 # WOPI discovery URL
 ProxyPass           /hosting/discovery https://127.0.0.1:9980/hosting/discovery
retry=0
 ProxyPassReverse    /hosting/discovery https://127.0.0.1:9980/hosting/discovery


 # Capabilities
 ProxyPass           /hosting/capabilities
https://127.0.0.1:9980/hosting/capabilities retry=0
 ProxyPassReverse    /hosting/capabilities
```

```
https://127.0.0.1:9980/hosting/capabilities

 # Main websocket
 ProxyPassMatch      "/cool/(.*)/ws$"      wss://127.0.0.1:9980/cool/$1/ws nocanon


 # Admin Console websocket
 ProxyPass           /cool/adminws wss://127.0.0.1:9980/cool/adminws


 # Download as, Fullscreen presentation and Image upload operations
 ProxyPass           /cool https://127.0.0.1:9980/cool
 ProxyPassReverse    /cool https://127.0.0.1:9980/cool
 # Compatibility with integrations that use the /lool/convert-to endpoint
 ProxyPass           /lool https://127.0.0.1:9980/cool
 ProxyPassReverse    /lool https://127.0.0.1:9980/cool
```

Reverse proxy settings in Apache2 config (SSL termination)

These lines should be inserted into `<VirtualHost>` definition of the site. Basically the configuration is the same as above, but in this case we have HTTP-only connection between the proxy and the Collabora Online server.

In `coolwsd.xml` the corresponding setting is `ssl.enable=false` and `ssl.termination=true`.

```
 ########################################

 # Reverse proxy for Collabora Online   #

 ########################################


 AllowEncodedSlashes NoDecode
 ProxyPreserveHost On


 # static html, js, images, etc. served from coolwsd
 # browser is the client part of Collabora Online
 ProxyPass           /browser http://127.0.0.1:9980/browser retry=0
 ProxyPassReverse    /browser http://127.0.0.1:9980/browser


 # WOPI discovery URL
 ProxyPass           /hosting/discovery http://127.0.0.1:9980/hosting/discovery
retry=0
 ProxyPassReverse    /hosting/discovery http://127.0.0.1:9980/hosting/discovery


 # Capabilities
 ProxyPass           /hosting/capabilities
http://127.0.0.1:9980/hosting/capabilities retry=0
 ProxyPassReverse    /hosting/capabilities
http://127.0.0.1:9980/hosting/capabilities


 # Main websocket
 ProxyPassMatch      "/cool/(.*)/ws$"      ws://127.0.0.1:9980/cool/$1/ws nocanon


 # Admin Console websocket
 ProxyPass           /cool/adminws ws://127.0.0.1:9980/cool/adminws


 # Download as, Fullscreen presentation and Image upload operations
```

```
ProxyPass           /cool http://127.0.0.1:9980/cool
ProxyPassReverse    /cool http://127.0.0.1:9980/cool
# Compatibility with integrations that use the /lool/convert-to endpoint
ProxyPass           /lool http://127.0.0.1:9980/cool
ProxyPassReverse    /lool http://127.0.0.1:9980/cool
```

2.8.2 Reverse proxy with Nginx webserver

Reverse proxy settings in Nginx config (SSL)

Add a new `server` block to your Nginx config for `collaboraonline.example.com`.

In `coolwsd.xml` the corresponding setting is `ssl.enable=true`.

```
server {
 listen       443 ssl;
 server_name  collaboraonline.example.com;


 ssl_certificate /path/to/certificate;
 ssl_certificate_key /path/to/key;


 # static files
 location ^~ /browser {
   proxy_pass https://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # WOPI discovery URL
 location ^~ /hosting/discovery {
   proxy_pass https://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # Capabilities
 location ^~ /hosting/capabilities {
   proxy_pass https://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # main websocket
 location ~ ^/cool/(.*)/ws$ {
   proxy_pass https://127.0.0.1:9980;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection "Upgrade";
   proxy_set_header Host $http_host;
   proxy_read_timeout 36000s;
 }


 # download, presentation and image upload
 location ~ ^/(c|l)ool {
   proxy_pass https://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # Admin Console websocket
 location ^~ /cool/adminws {
```

```
    proxy_pass https://127.0.0.1:9980;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header Host $http_host;
    proxy_read_timeout 36000s;
  }
}
```

Reverse proxy settings in Nginx config (SSL termination)

Add a new `server` block to your Nginx config for `collaboraonline.example.com`. Basically the configuration is the same as above, but in this case we have HTTP-only connection between the proxy and the Collabora Online server.

In `coolwsd.xml` the corresponding setting is `ssl.enable=false` and `ssl.termination=true`.

```
server {
 listen       443 ssl;
 server_name  collaboraonline.example.com;


 ssl_certificate /path/to/certificate;
 ssl_certificate_key /path/to/key;


 # static files
 location ^~ /browser {
   proxy_pass http://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # WOPI discovery URL
 location ^~ /hosting/discovery {
   proxy_pass http://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # Capabilities
 location ^~ /hosting/capabilities {
   proxy_pass http://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }


 # main websocket
 location ~ ^/cool/(.*)/ws$ {
   proxy_pass http://127.0.0.1:9980;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection "Upgrade";
   proxy_set_header Host $http_host;
   proxy_read_timeout 36000s;
 }


 # download, presentation and image upload
 location ~ ^/(c|l)ool {
   proxy_pass http://127.0.0.1:9980;
   proxy_set_header Host $http_host;
 }
```

```
 # Admin Console websocket
 location ^~ /cool/adminws {
   proxy_pass http://127.0.0.1:9980;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection "Upgrade";
   proxy_set_header Host $http_host;
   proxy_read_timeout 36000s;
 }
}
```

2.8.3 Load balancing

For collaborative editing to function correctly, it is vital to ensure that all users editing the same document end up being served by the same Collabora Office instance. Using the WOPI protocol, the https URL includes a unique identifier (WOPISrc) for use with this document. Thus load balancing can be done by using WOPISrc – ensuring that all URLs that contain the same WOPISrc are sent to the same Collabora Office instance.

Note: for optimal performance all load balanced nodes must run the same version of Collabora Online. Currently Javascript, CSS and HTML that is served contains a unique version specific hash to enable browser caching while ensuring consistent upgrades. This version is provided in URLs provided from discovery.xml. When doing an incremental upgrade of a cluster an upgraded node will still provide new Javascript for an old version hash but will avoid sending ETag and CacheControl headers so that the files will be re-loaded when next fetched. This ensures that many minor upgrades can be done incrementally while an HA cluster continues running.

Example with HAProxy

In this example we will do load balancing between two Collabora Online server instances, which are running in docker containers. Load balancing is based on `WOPISrc` URL parameter.

The browser reaches the proxy with HTTPS protocol. The proxy terminates the HTTPS connection and passes traffic to backends via HTTP. Therefore in Collabora Online's config file, in `/etc/coolwsd/coolwsd.xml` , or in the command line which starts `coolwsd` daemon, SSL should be disabled, and SSL termination should be enabled.

Listing 2.6. add the following blocks to /etc/haproxy/haproxy.cfg

```
frontend coolwsd
  bind *:443 ssl crt /path/to/your/certificate_and_key.pem
  mode http
  default_backend coolwsd
backend coolwsd
  timeout tunnel 3600s
  mode http
  balance url_param WOPISrc check_post
  hash-type consistent
  server coolwsd01 127.0.0.1:9993
  server coolwsd02 127.0.0.1:9994
```

Start Docker containers as described above, with `-p 127.0.0.1:9993:9980` and `-p 127.0.0.1:9994:9980`.

Example with Nginx

Just like in the previous section (`HAProxy`), the Nginx load balancer also utilizes the `WOPISrc` URL parameter. In this example SSL settings are managed by `Certbot` (see https://letsencrypt.org/). The load balancer server listens on standard HTTPS port 443, and HTTP port 80 is redirected to HTTPS port 443. The coolwsd servers are reached through port 9980 directly (private network). The address for the outside world (for WOPI hosts) is coolwsd.public.example.com.

```
upstream coolwsd {
  zone coolwsd 64k;
```

```
  hash $arg_WOPISrc;

  server coolwsd1.private:9980;
  server coolwsd2.private:9980;
}

server {
  listen 80 default_server;
  listen 443 ssl; # managed by Certbot
  ssl_certificate
/etc/letsencrypt/live/1b255632-ce4b-4581-9e80-16f701c27034.pub.cloud.scaleway.com/fullchain.pem;
 # managed by Certbot
  ssl_certificate_key
/etc/letsencrypt/live/1b255632-ce4b-4581-9e80-16f701c27034.pub.cloud.scaleway.com/privkey.pem;
 # managed by Certbot
  include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

  if ($scheme != "https") {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  server_name coolwsd.public.example.com;

  location / {
    proxy_pass                  http://coolwsd;
    proxy_set_header            Host $host;
    proxy_http_version          1.1;
    proxy_set_header            Upgrade $http_upgrade;
    proxy_set_header            Connection "upgrade";
    client_max_body_size        0;
  }
}
```

2.8.4 robots.txt

When you use Collabora Online behind a reverse proxy, add `Disallow: /browser/*` to your robots.txt file.

Chapter 3

Troubleshooting

3.1 Symptom-based Troubleshooting

3.1.1 Service Unavailable (error 503)

When you see this error after trying to load a document, it is possible, that Collabora Online service (coolwsd) does not run properly.

First thing to try on the Collabora Online host from the administrator's (root) command line (commands listed for systemd):

```
systemctl status coolwsd
```

If the status is not active, and error is indicated, then check the logs:

```
journalctl -r -u coolwsd
```

The default log level is **warning**. If the root cause of the error is not clear, you can set the log level to be more detailed. Edit /etc/coolwsd/coolwsd.xml configuration file, and set the log level e.g. to **trace**. Then restart the service with:

```
systemctl restart coolwsd
```

and check the log again (reset log level afterwards, as verbose logging has performance implications).

The most common problem is **forgetting the SSL setup**. The path to SSL certificate, CA certificate and private key must be valid in /etc/coolwsd/coolwsd.xml. They can be set under ssl as cert_file_path, ca_file_path and key_file_path respectively.

Exit code 70 from coolwsd means internal software error, and this is most likely an issue with accessing certificates. Verify that certificates are set properly in the configuration file, and the files themselves are available with the correct access rights, i.e. readable by the cool user.

3.1.2 Gray document area, no document loaded

When you see gray document are, i.e. the iframe of Collabora Online is not loaded, you can check the browser's console to see what is going on. For the browser's diagnostic tools, press **F12**, and see network activity.

A common mistake, when http and https content is mixed on the same page, and the browser refuses to load insecure http iframe into an https page. If SSL is used, which is highly recommended, make sure that all components, including Collabora Online is set up for SSL.

The iframe can be blocked due to a content security policy setting. The problem should be reported on the browser's console.

Gray screen could be the result of incorrectly set up reverse proxy (see below), and related server_name setting.

3.1.3 Network connectivity problems

Collabora Online host, document storage host and user's browser have to see each other all times. To this end, it is always recommended to set up a **reverse proxy** on Collabora Online host, because the default port of Collabora Online, port 9980 is sometimes blocked by users' corporate firewall, only standard https port 443 is allowed.

If the reverse proxy is not preserving the Collabora Online host name, it has to be set in server_name setting.

Preferably do not use non-routable internal IP addresses or domain names that DNS cannot resolve on all hosts (otherwise they have to be present in the /etc/hosts file).

A command line example: `coolconfig set server_name office.example.com`.

3.1.4 Load balancing problems

In incorrectly configured load balanced multi-node Collabora Online cluster you can experience different characteristic symptoms:

Pasting into document doesn't work. The browser's request gets response 400 Bad Request and in the server logs you can find: `ERR Cluster configuration error: mis-matching serverid AAAAAAA vs. BBBBBBB`

When two users join to the same document but doesn't see each other edits or saved content is lost or overwritten by other users working on the same file.

That means the HTTP requests related to the same file are not directed to the same Collabora Online node.

Read more at Section 2.8.3.

3.1.5 'Unauthorized WOPI host error' when opening a file.

The document storage host is not among the allowed WOPI hosts in /etc/coolwsd/coolwsd.xml configuration file (under element wopi). Add the domain name or the IP address, and restart coolwsd.

3.2 'External data source not allowed' when trying to insert content (image, clipboard paste).

We block requests from the Collabora Online server to the hosts which are not trusted. If you want to enable access to some hosts then you have to add regex into coolwsd.xml file in the``net.lok_allow`` section (host without port or scheme, IP or domain name depending on which one will be used in the request, we don't resolve names). Hosts already defined in `storage.wopi.alias_groups` or `net.post_allow` should be enabled automatically.

3.3 Further symptoms

3.3.1 Tiles load slowly or missing when working with a file.

The document area is rendered in tiles on the server, and those tiles are transferred to the client. Slow tile display is likely caused by network latency issues, however if tiles do not load at all sometimes, then a ticket should be opened.

3.3.2 File does not display correctly or editing actions do not give expected result.

The cause is likely a bug in the document rendering/editing layer, a ticket should be opened.

3.3.3 Issues that were supposed to be fixed in the current version are present.

The installation might be outdated, verify the version using Help → About or in the console log accessible via `journalctl -r -u coolwsd`. If the version is as expected, a ticket should be opened.

3.4 Package upgrade issues

3.4.1 Webserver restart

The WOPI discovery.xml file may be cached at WOPI host. It contains versioned URLs that will point to a non-existing location after coolwsd upgrade. The symptom is that the content is not loaded into the iframe, and there are error message in the coolwsd log file, related to missing files. In this case restart the web server.

3.5 Diagnostic capabilities

- Opening http(s)://<host>:<port>/ (as set up) in browser shows "OK" if service is running correctly
- Opening http(s)://<host>:<port>/hosting/discovery (as set up) in browser brings up the WOPI discovery file
- The following command brings up logs for the web socket daemon service: `journalctl -r -u coolwsd`
  Preferably set log level to trace in /etc/coolwsd/coolwsd.xml configuration file under 'logging'/'level' beforehand
- When opening a file, the browser's developer console logs potential errors and information related to the web page
- When a file is opened, Ctrl-Alt-Shift-D brings up a debug view that shows information related to rendered tiles, including server latency
- The admin console is accessible separately under the following location: http(s)://<host>:<port>/browser/dist/admin/admin.html

  It provides details on consumed memory, users online and documents open.
  Note that a user/password combination has to be set up for admin console with the following command: `coolconfig set-admin-password`

3.6 Case studies

3.6.1 Case study No. 1

A partner installed Collabora Online from the provided packages, and set up the configuration. However, upon starting, coolwsd service exited with an error.

The partner provided their configuration and logs (after setting log level to 'trace').

From the logs it was apparent the service exited with status code 70, and the error was also in the log file:

```
Error loading private key from file <file name> (error:0200100D:system
library:fopen:Permission denied).
```

After correcting file permissions, the service started successfully.

After this there was another issue, the WOPI host was missing from the /etc/coolwsd/coolwsd.xml configuration file. After adding host name to 'wopi' element the service was running and accessible.

3.6.2 Case study No. 2

A partner installed Collabora Online from the provided packages, and set up the configuration to access files locally (enabled setting storage / filesystem), but was not able to access files. In a different, container-based setup, accessing local files worked.

Browser console logs showed that the server was accessed without the required port, and the requested file was not found. At first the reverse proxy seemed to be the culprit, but the issue persisted without proxy as well.

The /hosting/discovery file already listed the target host and path without the port. This pointed to the 'server_name' setting in /etc/coolwsd/coolwsd.xml, which was set, but without the necessary port. However in this case the setting was not necessary at all, and clearing it fixed the issue.

Chapter 4

Introduction

This document will help you integrating with your existing solution (file storage) so that your users can edit the documents hosted in the file storage via web browser.

For this to work, you have to setup and connect several parts together:

- Server for hosting Collabora Online
- Website that presents iframe with the editing capabilities
- Authentication
- Connection to the file storage

The easiest way to integrate Collabora Online is using the WOPI protocol.

Collabora Online implements a protocol inspired by the WOPI (Web Application Open Platform Interface) protocol. Using the WOPI vocabulary as we do throughout this document, Collabora Online is a *WOPI client* that can be integrated with a *WOPI host* (your existing solution: web application and file storage). WOPI is a well documented open protocol, for more details please visit the WOPI documentation.

Chapter 5

How to integrate

5.1 Server for hosting Collabora Online

Although it is possible to host Collabora Online on the same server where you run your web services, we strongly recommend a dedicated VM or server for that. Such a server has to be accessible from the Internet, and has to be able to connect to the server running your document storage.

5.2 Website that presents the editing capabilities

We assume that you want to integrate the editing capabilities into your existing website. On the website, you need to present an iframe where the editing UI and the document itself will be present.

To set up the iframe, the *WOPI host* (your application) needs to read a discovery XML from a defined location on the *WOPI client* (the Collabora Online server). The discovery is available at:

```
https://<WOPI client URL>:<port>/hosting/discovery
```

The reply is discovery.xml that contains *urlsrc* for various file formats. The urlsrc specifies the address that you need to use for the iframe that you create for the document editing, and is set as an attribute of the HTML and or tag of the document.

You also need to pass the authentication token to Collabora Online via a form post, and the actual URL that your file storage can accept. The URL should look like:

```
https://<WOPI host URL>/<...>/wopi/files/<id>
```

Here `/wopi/` can actually be any string that starts with `wopi`, like `/wopifiles/` or `/wopi_implementation/`, but for simplicity, we will be using only `/wopi/` in the following text.

`<id>` should be URL-safe base64 (base64url) encoded, ie. only contain letters (`A-Z`, `a-z`), numerals (`0-9`), and - and _ symbols.

5.3 Authentication

To be able to access files securely, your application has to pass an authentication token to Collabora Online `access_token`. From the Collabora Online point of view, it can be any random number / string, that will be passed as part of the URL when accessing the document storage.

The only requirements are that it has to be unique for the user, that the file storage denies access with wrong authentication token, and that it can be passed in an URL (ie. contains just characters / numbers / underlines).

Currently this is the only supported way of authentication (`access_header` has been deprecated).

5.4 Connection to the file storage

As the *WOPI host*, your existing solution has to implement few entry points for Collabora Online (the *WOPI client*), so that Collabora Online can download files that the user wants to edit, and upload back the updates.

The *WOPI client* (Collabora Online) invokes the WOPI url created above to download the file:

```
GET https://<WOPI host URL>/<...>/wopi/files/<id>/contents?access_token=<token>
```

And to upload a file:

```
POST https://<WOPI host URL>/<...>/wopi/files/<id>/contents?access_token=<token>
```

Currently, Collabora Online only depends on WOPI File Operation functions (GetFile/PutFile/CheckFileInfo). As a bare minimum, your application has to support the following four functions:

1. A function that generates a token for a given file and user (probably you want to store that in a DB, optionally with expiration).
2. **GetFile** that sends back the content of the file when the

   ```
   https://<WOPI host URL>/<...>/wopi/files/<id>/contents?access_token=<token>
   ```

3. **PutFile** that replaces the file with the body of the POST verb when invoked with the

   ```
   https://<WOPI host URL>/<...>/wopi/files/<id>/contents?access_token=<token>
   ```

   URL

   - An optional LastModifiedTime field, containing the ISO8601 round-trip time format indicating the new/updated file's modified time in storage after successful save, can be added to the JSON response. See for more details.

4. **PutRelativeFile** that creates a new file with the body of the POST verb for the needs of the Save As operation when invoked with

   ```
   https://<WOPI host URL>/<...>/wopi/files/<id>?access_token=<token>
   ```

   URL. If you do not want to support the Save As operation, please add **UserCanNotWriteRelative** with value **true** to your CheckFileInfo answer. It is request header X-WOPI-SuggestedTarget that is supported.

5. **CheckFileInfo** that returns (at least) the BaseFileName and Size of the file as json when the

   ```
   https://<WOPI host URL>/<...>/wopi/files/<id>?access_token=<token>
   ```

   URL is invoked. Collabora Online takes the following required CheckFileInfo properties:

   - `BaseFileName` – The string name of the file without a path. Used for display in user interface (UI), and determining the extension of the file.
   - `OwnerId` – A string that uniquely identifies the owner of the file.
   - `Size` – The size of the file in bytes, expressed as a long, a 64-bit signed integer.
   - `UserId` – A string value uniquely identifying the user currently accessing the file.

   Collabora Online takes the following optional CheckFileInfo properties:

   - `UserFriendlyName` – The name of the user, suitable for displaying on the UI.
   - `UserCanWrite` – It has to be set to true if the file is opened for editing.
   - `PostMessageOrigin` – It is used by PostMessage API.

- `HidePrintOption` – Hides print button from UI but accessible using PostMessage API so hosts can implement their own UI for this.
- `DisablePrint` – Disables printing of documents. Additionally, hides print option from UI.
- `HideSaveOption` – Hides save button from UI. Manual save can still triggered using PostMessage API. Does not affect automatic save.
- `HideExportOption` – Hides 'Download as' button/menubar item from UI. Can still be triggered using PostMessage API.
- `DisableExport` – Disable export of the document in any format. Additionally, hides the 'Download as' button from the UI.
- `DisableCopy` – Disable copying from the document.
- `EnableOwnerTermination` – This gives document owners the ability to terminate all sessions currently having that document opened.
- `LastModifiedTime` – ISO8601 round-trip time format for file's last modified time in storage.
- `IsUserLocked` – Lock the user from using certain feature(s) which can be later be unlock by user.
- `IsUserRestricted` – Disable feature(s) for the user.

5.5 Re-using our development / demo-servers

One easy way to test your WOPI integration without even needing to setup Collabora Online for both development, and your users is to target one of our demo servers read how to do that properly you can provide users' a list of servers from this end-point however please bear in mind these factors:

- you must have a publicly routable and resolvable WOPI server. If you pass our demo servers a WOPI URL to 192.168.1.5 we will not be able to get to the document data (obviously) to load, save and render it. Of course, it can be rather easier to trace protocol problems in the logs of your own server too.
- it is vitally important to let the user know that their data will be shared with others and that they should not include personally identifying information into their test documents.
- it is worth reminding users that the performance of such a shared test server can be extremely variable and is not representative of a properly setup on-premise installation.

5.6 Further differences to WOPI

In addition to the basics of WOPI as described in WOPI specifications, Collabora Online implements various extensions, in addition to those outlined above primarily associated with CheckFileInfo, to support some features that you may find useful.

1. **Custom Button API**: It is possible to add your own custom buttons to the editor's UI. For more information, you can check insert_button and clicked_button API PostMessage API.
2. **Detecting external document change**: In some cases, when the document is updated in your storage while being edited in Collabora Online and there are unsaved changes, we detect it as soon as possible and ask the user if he/she would like to overwrite the changes or reload the new document from the storage.
   In case there are no unsaved changes, we reload the new document without asking the user.
   To be able to support this feature, you need to specify LastModifiedTime field in both CheckFile-

Info and PutFile calls as described above in its documentation.
Additionally, WOPI hosts must check for a header in PutFile response – `X-COOL-WOPI-Timestamp`. This header contains the ISO8601 round-trip formatted time of file's last modified time in storage, as known by Collabora Online. In case, this header is present and its value doesn't match with the file's modified time in storage, it indicates that document being edited is not the one that is present in the storage. WOPI hosts should not save the file to storage in such cases and respond with HTTP 409 along with Collabora Online specific status code for this purpose in JSON response against the field *COOLStatusCode*. The COOLStatusCode for this purpose is 1010. So, the desired response should be:

Listing 5.1. HTTP 409 with JSON

```
{
    'COOLStatusCode': 1010
}
```

Step-by-step tutorial

It is not practical trying to implement everything in one go. We recommend building the integration in small, easily testable steps, like the following:

1. Install Collabora Online on a dedicated server or in a VM, and make sure you can access the discovery service by pointing your browser to

   ```
   https://<WOPI client URL>:<port>/hosting/discovery
   ```

2. Add WOPI REST endpoints to your web service, for the moment returning only a "Hello World" message upon a GET request, that you can evaluate via a web browser. If you need, you can for example use Apache's mod_rewrite so that the REST endpoints are redirected to URL of your choice. At this moment, test that

   ```
   https://<WOPI host URL>/<...>/wopi/files/<id>/contents
   ```

   returns Hello World for whatever `<id>`.

3. Implement the CheckFileInfo endpoint – make sure that

   ```
   https://<WOPI host URL>/<...>/wopi/files/<id>
   ```

   returns JSON like `{ "BaseFileName": "test.txt", "Size": 11 }`

4. At this moment, you will be able to see a constructed document in Collabora Online: Create a URL by concatenating URL from the discovery XML (see the point 1), and add `WOPISrc=https://<WOPI host URL>/<...>/wopi/files/<id>` at the end, resulting in URL like

   ```
   https://<WOPI client
   URL>:<port>/browser/<hash>/cool.html?WOPISrc=https://<WOPI host
   URL>/<...>/wopi/files/<id>
   ```

   Create a small test.html file containing:

   ```
   <html><body>
           <form action="...URL you constructed..." enctype="multipart/form-data"
    method="post">
           <input name="access_token" value="test" type="hidden"/>
           <input type="submit" value="Load Collabora Online"/>
           </form>
   </body></html>
   ```

   When you load it in the browser and click the **Load Collabora Online** button, it will open a text document that shows "Hello World" provided by the WOPI endpoints. If your WOPI host is on a different machine than Collabora's, make sure to add that host along with the port in the configuration file coolwsd.xml under `<storage>` → `<wopi>` tag.

5. From this point, you have the basics working, and you have to extend the JavaScript pieces: Create an iframe that will contain the Collabora Online, and provide that with a real access

token. From the Collabora Online point of view, the access token can be any random text or number that contains just numbers, characters, and underlines.

6. Update your REST endpoints so that they provide real data instead of a synthesized "Hello World" and hard-coded document length.
7. Implement the PutFile end point, so that the results of editing can appear in your file storage too. To do this, implement the POST request to

```
https://<WOPI host URL>/<...>/wopi/files/<id>/contents
```

8. Either implement the PutRelativeFile endpoint, so that the option to Save As appears in the UI, or change CheckFileInfo to return **UserCanNotWriteRelative** with value **true** if you don't want to do that yet.

Simple examples

Various integration examples using different programming languages. Minimal in complexity these can be a great start in point for your integration. Feel free to use and remix these at your will.

7.1 Node.js example

A simple example integrating Collabora Online via `iFrame` in Node.js. We assume you are familiar with npm and the node.js framework.

Node.js Example on GitHub

7.2 PHP example

A simple example integrating Collabora Online via `iFrame` in PHP. We assume that you have already the Apache web server installed on your machine and started, and that the PHP module for Apache has been installed too and loaded.

PHP Example on GitHub

7.3 Python example

A simple example integrating Collabora Online via `iFrame` in Python. We assume you are familiar with Python and Django framework.

Python Example on GitHub

7.4 ReactJS example

A simple example integrating Collabora Online via `iFrame` in ReactJS and Express. We assume you are familiar with ReactJS, npm and node.js framework.

ReactJS Example on GitHub

7.5 .NET example

A simple example integrating Collabora Online via `iFrame` with .NET backend.

.NET Example on GitHub

More examples in various other languages are coming soon. In the meantime, you can investigate the full production-grade integrations.

Chapter 8

Available integrations

8.1 Alfresco integration

Collabora Online makes a great combination with the Alfresco open source enterprise management solution, when integrated with the connector by *Jeci <https://jeci.fr>*, a partner of Collabora.

This Alfresco Community integration allows you to edit documents stored in Alfresco in the browser (text documents, spreadsheets and presentations) simultaneously on both Alfresco Share and Alfresco Content App.The extension adds an action Edit with Collabora™ Online on documents which can be opened with Collabora Online and if the user has the write permission. The document will be opened in an `iFrame`.

Listing 8.1. `alfresco-collabora-online.git`

```
git clone https://github.com/CollaboraOnline/alfresco-collabora-online.git
```

More info on Alfresco integration

8.2 EGroupware integration

The Collabora Online integration enables EGroupware users to create, open and edit office files, such as text documents, spreadsheets or presentations, directly from within EGroupware. The integration adds a lot of convenient features to your collaborative workflow. It enables you to email documents directly from Collabora Online. You can also use your own default documents and insert images using EGroupware's native file manager. EGroupware offers both an on-premise and a cloud solution.

Listing 8.2. `collabora.git`

```
git clone https://github.com/EGroupware/collabora.git
```

More Info on EGroupware integration

8.3 Mattermost integration

Collabora Mattermost integration allows Mattermost users to view and edit files directly in Mattermost. The integration will parse the messages that contain attachments and will display a list with the files that can be viewed or edited. The file is automatically saved when editing, so you don't have to do it manually.

Listing 8.3. `collabora-mattermost.git`

```
git clone https://github.com/CollaboraOnline/collabora-mattermost.git
```

More Info on Mattermost integration

8.4 Moodle integration

The Collabora Online plugin for Moodle brings new possibilities to the popular open source learning platform: live editing of and cooperating on office documents. Read more.

8.4.1 Collaborative document editing

This activity module enables Moodle users to create documents (simple text files, rich text, spreadsheets and presentation documents or upload a document) via a self-hosted Collabora Online Server and work collaboratively on this documents.

Listing 8.4. `moodle-mod_collabora.git`

```
git clone https://github.com/learnweb/moodle-mod_collabora.git
```

8.4.2 Collaborative submissions

Use live document collaboration within assignments, powered by Collabora Online.

This sub-module adds an additional type to `submissions` section: `Collaborative Submissions` and other options (initial file name, format, width and height of the document) allowing the student to start a submission by creating a new document on the fly.

Listing 8.5. `moodle-assignsubmission_collabora.git`

```
git clone https://github.com/CollaboraOnline/collabora-mattermost.git
```

8.5 Nextcloud integration

Nextcloud offers a unique-in-the-industry fully open source solution for on-premise data handling and communication with a uncompromising focus on security and privacy. The `richdocumentscode` is a built-in server with all of the document editing features of Collabora Online. Easy to install, for personal use or for small teams. A bit slower than a standalone server and without the advanced scalability features.

Listing 8.6. `richdocumentscode.git`

```
git clone https://github.com/CollaboraOnline/richdocumentscode.git
```

More Info on Nextcloud integration

8.6 ownCloud integration

Collabora Productivity and ownCloud are proud to release a combined commercial solution including Collabora Online. Key features include Collaborative editing; Author new content; View and edit documents, spreadsheets and presentations; Preservation of layout and formatting of documents; Insert comments, reply to comments; File revision history; Full screen presentation; Support for any modern web browser; Increase productivity while you stay in full control of sensitive corporate data

Listing 8.7. `richdocumentscode-owncloud.git`

```
git clone https://github.com/CollaboraOnline/richdocumentscode-owncloud.git
```

More Info on ownCloud integration

8.7 SharePoint integration

Collabora Online is perfectly equipped to integrate with your SharePoint 2016 Server. With just a few tweaks in your settings you will be able to open and edit documents using Collabora Online.

How to WOPI binding for Collabora Online in Sharepoint

More Info on SharePoint integration

Chapter 9

Advanced integration

Collabora Online uses a WOPI-like protocol to interact with hosts who want to integrate Collabora Online in them.

Refer to WOPI docs for further details on the protocol's inspiration.

9.1 CheckFileInfo response properties

9.1.1 BaseFileName

A string containing the base name of the file, omitting its path.

9.1.2 DisablePrint

Disables print functionality in Collabora Online (COOL) backend. If true, HidePrintOption is assumed to be true.

9.1.3 OwnerID

A programmatic string identifier for the owner of the file.

9.1.4 PostMessageOrigin

A string for the domain the host page sends/receives PostMessages from, we only listen to messages from this domain.

9.1.5 Size

Size of the file in bytes (64bit)

9.1.6 TemplateSource

The ID of file (like the wopi/files/ID) can be a non-existing file. In that case, the file will be created from a template when the template (eg. an OTT file) is specified as `TemplateSource` in the CheckFileInfo response.

The TemplateSource is supposed to be an URL like `https://somewhere/accessible/file.ott` that is accessible by the Online.

For the actual saving of the content, normal PutFile mechanism will be used.

9.1.7 UserCanWrite

A boolean flag, indicating whether the user has permission to edit and/or over-write the file. If not set PutFile will fail.

9.1.8 UserCanNotWriteRelative

A boolean flag indicating that the user cannot Save-As on this server, so PutRelativeFile will fail.

9.1.9 UserId

A programmatic string identifier of the user.

9.1.10 UserFriendlyName

A string representing the name of the user for display in the UI.

While nominally an optional field, it is used to identify the author of changes in documents. When missing, `UnknownUser` will be used instead, with a possible suffix with the `UserId`.

Strongly recommended to set it to a valid value.

9.2 CheckFileInfo extended response properties

9.2.1 EnableInsertRemoteImage

If set to true, this will enable the insertion of images chosen from the WOPI storage. A UI_Insert-Graphic postMessage will be send to the WOPI host to request the UI to select the file.

9.2.2 DisableInsertLocalImage

If set to true, this will disable the insertion of image chosen from the local device. If `EnableInsertRemoteImage` is not set to true, then inserting images files is not possible.

9.2.3 HidePrintOption

If set to true, hides the print option from the file menu bar in the UI.

9.2.4 HideSaveOption

If set to true, hides the save button from the toolbar and file menubar in the UI.

9.2.5 HideExportOption

Hides `Download as` option in the file menubar.

9.2.6 DisableExport

Disables export functionality in backend. If set to true, HideExportOption is assumed to be true.

9.2.7 DisableCopy

Disables copying from the document in COOL backend. Pasting into the document would still be possible. However, it is still possible to do an “internal” cut/copy/paste.

9.2.8 DisableInactiveMessages

Disables displaying of the explanation text on the overlay when the document becomes inactive or killed. With this, the JS integration must provide the user with appropriate message when it gets `Session_Closed` or User_Idle postMessages.

9.2.9 DownloadAsPostMessage

Indicate that the integration wants to handle the downloading of pdf for printing or svg for

slideshows or exported document, because it cannot rely on browser's support for downloading.

When this is set to true, the user's eg. Print action will trigger a postMessage called Download_As, with the following JSON in the Values:

```
{ Type: 'print'|'slideshow'|'export', URL: ...url you use for the actual
downloading... }
```

9.2.10 SaveAsPostmessage

Similar to download as, doctype extensions can be provided for save-as. In this case the new file is loaded in the integration instead of downloaded.

```
{format: '<extension>' }
```

To achieve this, `args: {format: '<extension>' }` parameter needs to be sent inside UI_SaveAs. The integration should provide dialog with filename, there the extension will be set after the filename. The remaining work is already handled by save-as.

9.2.11 EnableOwnerTermination

If set to true, it allows the document owner (the one with `OwnerId =UserId`) to send a `closedocument` message. For more information please visit protocol.txt.

9.2.12 UserExtraInfo

JSON object that contains additional info about the user, namely the avatar image.

Listing 9.1. Example: User's additional info

```
{
  'avatar': 'http://url/to/user/avatar',
  'mail': 'user@server.com'
}
```

Listing 9.2. Example: User's additional info via PHP

```
'UserExtraInfo' => [ 'avatar' => 'http://url/to/user/avatar', 'mail' =>
'user@server.com' ]
```

Note: There is strict Content Security Policy that restricts image resources (`img-src`), therefore the avatar URL must not violate the CSP, otherwise it will show as broken images.

9.2.13 UserPrivateInfo

JSON object that contains additional info about the user, but unlike the `UserExtraInfo` it is not shared among the views in collaborative editing sessions.

For example it can hold the `ZoteroAPIKey` which is the personal API key to Zotero Web API, for working with citation databases. When the integration provides a Zotero API key for the user, the functionality of handling Zotero databases gets enabled in Collabora Online's user interface.

9.2.14 WatermarkText

If set to a non-empty string, is used for rendering a watermark-like text on each tile of the document.

Note: It is possible to just hide print, save, export options while still being able to access them from other hosts using PostMessage API.

9.3 PostMessage extensions

9.3.1 App_LoadingStatus

Was extended with field 'Status' with `Document_Loaded` value when document was loaded successfully and 'Failed' in other case.

9.4 PutFile headers

PutFile additionally indicates whether the user has modified the document before the save, or if they just pressed the Save button without any modification. The following header:

```
X-COOL-WOPI-IsModifiedByUser
```

will have the value `true` or `false` accordingly.

To distinguish auto-save vs. explicit user requests to save, the following header:

```
X-COOL-WOPI-IsAutosave
```

will have the value `true` when the PutFile is triggered by auto-save, and `false` when triggered by explicit user operation (Save button or menu entry).

When the document gets cleaned up from memory (e.g. when all users disconnect), an automatic save will be triggered. In this case the following header will be set to `true`:

```
X-COOL-WOPI-IsExitSave
```

9.5 Detecting external document change

Locking is omitted from our WOPI-like protocol since it goes against common EFSS solutions usage. Instead, Collabora Online uses timestamps to detect document changes.

When the document is updated in your storage while being edited in Collabora Online and there are unsaved changes, we detect it as soon as possible and ask the user if he/she would like to overwrite the changes or reload the new document from the storage.

In case there are no unsaved changes, we reload the new document without asking the user.

To support this feature, the host implementation has to specify LastModifiedTime field in both CheckFileInfo and PutFile calls.

Additionally, hosts must check for a header in PutFile response:

```
X-COOL-WOPI-Timestamp
```

This header contains the ISO8601 round-trip formatted time of file's last modified time in storage, as known by Collabora Online. In case this header is present and its value does not match the file's modified time in storage, it indicates that document being edited is not the one that is present in the storage.

Hosts should not save the file to storage in such cases and respond with HTTP 409 along with Collabora Online specific status code:

Listing 9.3. HTTP 409 with JSON

```
{
```

```
    'COOLStatusCode': 1010
}
```

When the user chooses "overwrite" when asked how to resolve the conflict, Collabora Online will attempt one more save operation, but this time it will lack the `X-COOL-WOPI-Timestamp` header, which means "save regardless of state of the file".

9.6 Checking for available features

With new features, it is important for the integrations to know if the Online they are using is supporting them. For this reason, we have introduced a `/hosting/capabilities` endpoint that returns a JSON with information about the availability of various features.

Currently the following are present:

- `convert-to: { available: true/false }`
  The property `available` is `true` when the convert-to functionality is present, and is allowed from the requesting address. Because of that, the endpoint needs to be accessed from the WOPI host for it to return relevant result.
  A common use of the functionality is that WOPI hosts can use it to generate document previews to show in their file list.
- `hasTemplateSource: true/false`
  is `true` when Collabora Online supports the `TemplateSource` `CheckFileInfo` property.
- `hasMobileSupport: true/false`
  is `true` when Collabora Online has a good support for the mobile devices and responsive design.

9.7 Modifying discovery.xml

The discovery.xml defines which part of Collabora Online can be used for a certain file type and what action should be used.

The following is a snippet from discovery.xml:

```
...
<app name="writer" favIconUrl="images/x-office-document.svg">
   <action name="view" default="true" ext="sxw"/>
   <action name="edit" default="true" ext="odt"/>
   ...
</app>
...
<app name="draw">
   ...
   <action name="view_comment" ext="pdf"/>
</app>
```

The supported actions are "edit", "view" and a special "view_comment". When the action is "edit", the document will open for editing. When the action is "view", the document will open in read-only mode for viewing. When the action is set to "view_comment", the document will open read-only, but adding or editing comments will still be possible.

The "view_comment" mode is useful for PDF documents, where adding and editing comments is allowed, but otherwise no other editing should be allowed to be performed.

9.8 Override CheckFileInfo

Section 9.2.9 property of the CheckFileInfo can be overridden. This is controlled by:

```
<input name="checkfileinfo_override" value="DownloadAsPostMessage=VALUE"
type="hidden"/>
```

during sending the form when the iframe is being set up (similarly as the access_token). The VALUE can be either true or false. This will override `DownloadAsPostMessage` value from CheckFileInfo response.

9.9 Clipboard handling

9.9.1 Allow the clipboard permission query

Chrome implements a JavaScript API to paste from the clipboard without using the keyboard. This relies on cross-origin iframes, which is disabled by default. To enable it in your integration, use a new attribute in the iframe element of your integration:

```
<iframe allow="clipboard-read *; clipboard-write *"/>
```

Once this is in place, the user will see a popup to allow access to the clipboard on first use.

9.9.2 Extensions to the HTML clipboard

HTML marker

When Collabora Online puts HTML to the clipboard, it puts a marker into the body, so it can recognize this content comes from itself and this allows performing a better internal copy:

```
<body>
    <div id="meta-origin" data-coolorigin="...">
        ... original body content ...
    </div>
</body>
```

With this, the copy of complex content like charts is performed in an improved way (on the same COOL instance or between COOL instances), better than what's possible via HTML export and HTML import.

Spreadsheet HTML extensions

Specific to spreadsheets, Collabora Online Calc implements the following HTML extensions to maintain better copy & paste when going via HTML:

- After the own Collabora Online marker, also a `<google-sheets-html-origin/>` marker is emitted to please Google Sheets.
- In case the value of a cell is text, and it should not be recognized as a number, that can be signalled explicitly:

```
<table>
    <tr>
        <td
data-sheets-value="{&quot;1&quot;:2,&quot;2&quot;:&quot;01&quot;}">01</td>
    </tr>
</table>
```

The `data-sheets-value` attribute on the `td` element has a JSON value, where the `1` key is set to `2` to say the type is text, and the `2` key contains the `01` text which won't be auto-converted to `1` on import.

- In case the value of a cell is boolean, the markup for that is:

```
<table>
    <tr>
        <td data-sheets-value="{&quot;1&quot;:4,&quot;4&quot;:1}">WAHR</td>
    </tr>
</table>
```

The `data-sheets-value` attribute on the `td` element has a JSON value, where the `1` key is set to `4` to say the type is boolean, and the `4` key contains 1, so even a localized `TRUE` value is recognized on export. `0` can be used to express `FALSE`.

- Formatted numbers can have metadata to express their float values:

```
<table>
    <tr>
        <td data-sheets-value="{&quot;1&quot;:3,&quot;3&quot;:1000}"
data-sheets-numberformat="{&quot;1&quot;:2,&quot;2&quot;:&quot;#,##0.00&quot;,&quot;3&quot;:1}">
    </tr>
</table>
```

The `data-sheets-value` attribute on the `td` element has a JSON value, where the `1` key is set to `3` to say the type is float, and the `3` key contains the original number. Additionally, the `data-sheets-numberformat` attribute also contains a JSON value. The `1` key is set to `2` to describe a number format, the `2` key contains the actual number format.

This allows copying not only the formatted number, but also the generator float value and its number format.

- In case the float number of a cell is a result of a formula, the original formula can be also described:

```
<table>
    <tr>
        <td data-sheets-value="{&quot;1&quot;:3,&quot;3&quot;:1}">1</td>
        <td data-sheets-value="{&quot;1&quot;:3,&quot;3&quot;:2}">2</td>
        <td data-sheets-value="{&quot;1&quot;:3,&quot;3&quot;:3}"
data-sheets-formula="=SUM(R[0]C[-2]:R[0]C[-1])">3</td>
    </tr>
</table>
```

The `data-sheets-formula` attribute on the `td` element contains the formula, grammar is R1C1 reference style.

- In case one or multiple cells are copied, then one `table` element, one or more `tr` elements and one or more `td` elements are to be used. If a single cell is copied, then the above `data-*` attributes can be also emitted on a `span` element:

```
<span data-sheets-value="{&quot;1&quot;:3,&quot;3&quot;:3}"
data-sheets-formula="=SUM(R[0]C[-2]:R[0]C[-1])">3</span>
```

This `<span>` syntax is read, but not written by Collabora Online.

Google Sheets seems to not publish a specification for their HTML extensions. We figured out, that the markup is something like the above. Comments regarding this markup are welcome on the `devel@documentliberation.org` mailing list.

9.10 URL query parameters

Some additional options can be passed using URL query parameters.

Those suported are:

- `closebutton`: allows to display a close button, that will emit the `UI_Close` message on the postMessage API, with the parameter `EverModified`, telling if the file had any modification. Example: `&closebutton=true`
- `revisionhistory`: allows to display the *See history* option in the *File* tab or *File* menu. This button, when clicked, emits the `UI_FileVersions` message on the postMessage API. Example: `&revisionhistory=true`
- `target`: allows to focus a section in the document opened upon loading. Example: `&target=image6.png|graphic`
- `timestamp`: allows to pass in the modification time of the document as a Unix timestamp that will be passed to the server.

And some debug and testing utilities:

- `lang`: allows to manually overwrite the language. Example `&lang=ar` or `&lang=fr`
- `permission`: allows to load a document in readonly mode. Example: `&permission=readonly`
- `debug`: enables debug mode, displaying controls useful to debug Collobora Online. Example: `&debug=true`, there is a shortcut Ctrl+Shift+Alt+D.
- `randomUser`: enables debug mode and loads a random language. Example: `&randomUser=true`

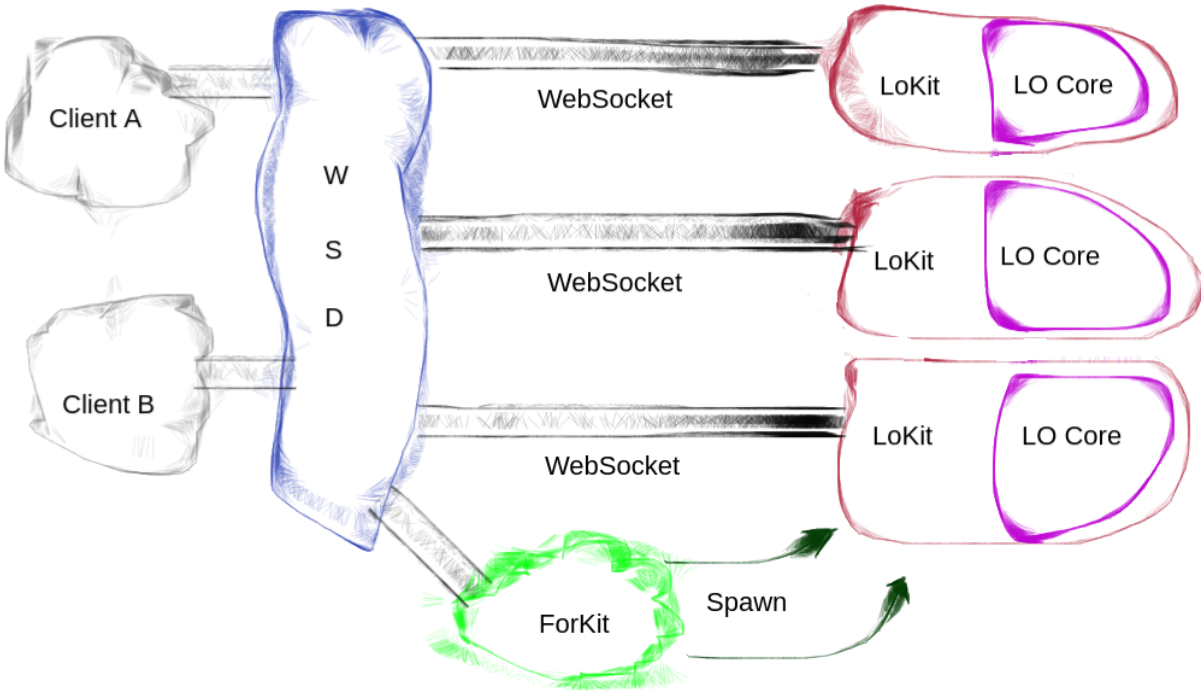
Architecture

Collabora Online is designed to be self-contained and secure out of the box. It has its own built-in web-server, which is often ran behind a reverse proxy for flexibility and added security, as well as isolated per-document processes.

Internally, each document is loaded in a separate process, which is isolated on multiple levels.

Conceptually, there are three (3) groups of processes: WSD, ForKit, and Kit.

A high-level sketch looks as follows:



10.1 WSD

The main process, WSD (Web Service Daemon), is responsible for spawning ForKit, setting up the childroot directory, and listening for incoming connections.

Once ready, the WSD process accepts incoming connections on the incoming port (9980 by default). This is the primary server connection which accepts client connection.

10.2 ForKit

The ForKit process is responsible for loading the Collabora Office libraries. These dynamic shared object(s), or DSO for short, implement the core logic of viewing and editing the document. This is referred to as the Core.

In addition to the core logic, these libraries also implement the API to communicate with WSD and therefore implement the interface with the Collabora Online. This layer is called the Kit.

Once the Kit and Core libraries are loaded, ForKit is ready to fork additional processes that can load documents. The ForKit process never loads documents, however. It is only responsible for forking additional processes to load documents, per WSD's demands.

The name 'ForKit', therefore, is a play on both 'fork it' and 'fork Kit' by contracting the latter into a single word.

10.3 Kit

The Kit process is forked from ForKit and is therefore by and large identical to it. Except, there are very important differences. Unlike ForKit, the Kit process is only responsible for loading documents and passing input and commands from users to the document and similarly for the events generated by the document back to the client code. Therefore, it doesn't fork any processes. Instead, it creates an isolated environment Before even becoming visible to WSD as available to load a document. See Kit Isolation below.

10.4 Kit Isolation

Before isolating the Kit process, it needs to create a shadow file system (see below for details). Preparing the shadow file system can be done either by bind-mount from a template directory (preferred) or by linking (if possible) or, failing that, by copying the files. The template directory contains important files to run the Core as well as various files from the system (from /etc) for the timezone, hosts, and similar files.

When bind-mount is enabled (default) and available (some systems and especially containers might not allow it), it is very fast to set up and is done with read-only permissions, which adds yet another layer of security.

After the shadow file system is ready, chroot(2) is called to change the visible root file system for the current Kit process to be that of the shadow file system. No files outside of this root is visible or accessible to the Kit process, isolating it completely from the host system.

Next, since certain capabilities are needed to execute the above (specifically, mount and chroot are privileged), capabilities are dropped to minimize the available system calls.

Finally, all system calls are disabled, even the unprivileged ones, with the exception of the strictly required ones. For example, the kill(2) sys- call is disabled, as well accept(2) and listen(2), which are used to create listening sockets.

10.5 Architecture

There are three processes: CoolWSD, CoolForKit, and CoolKit.

WSD is the top-level server and is intended to run as a service. It is responsible for spawning ForKit and listening on public port for client connections.

The ForKit is only responsible for forking Kit instances. There is only one ForKit per WSD instance

and there is one Kit instance per document.

WSD listens on a public port and using internal pipes requests the ForKit to fire a child (Kit) instance to host documents. The ForKit then has to find an existing Kit that hosts that document, based on the public URI as unique key, and forward the request to this existing Kit, which then loads a new view to the document.

There is a singleton Admin class that gets notified of all the important changes and update the AdminModel object accordingly. AdminModel object has subscribers which corresponds to admin panel sessions. Subscriber can subscribe to specific commands to get live notifications about, and to update the UI accordingly.

Whether a document is loaded for the first time, or this is a new view on an existing one, the Kit connects via a socket to WSD on an internal port. WSD acts as a bridge between the client and Kit by tunnelling the traffic between the two sockets (that which is between the client and WSD and the one between WSD and Kit).

10.6 File System

WSD is given childroot argument through config (child_root_path). This is the root directory of jailed FS. This path can be anywhere, but here we'll designate it as:

```
/childroot
```

Before spawning a ForKit instance, WSD needs to generate a random Jail-ID to use as the jail directory name. This `JailID` is then passed to ForKit as argument `jailid`.

Note: for security reasons, this directory name is randomly generated and should not be given out to the client. Since there is only one ForKit per WSD instance, there is also one `JailID` between them.

The ForKit creates a chroot in this directory (the jail directory):

```
/childroot/jailid/
```

ForKit copies the LO `instdir` (essentially installs LO in the chroot), then copies the Kit binary into the jail directory upon startup. Once done, it chroot-s and drops caps.

ForKit then waits on a read pipe to which WSD writes when a new request from a client is received. ForKit is responsible for spawning (or forking) Kit instances. For our purposes, it doesn't matter whether Kit is spawned or forked.

Every document is hosted by a Kit instance. Each document is stored in a dedicated directory within the jail directory. The document root within the jail is /user/docs. The absolute path on the system (which isn't accessible to the Kit process as it's jailed) is:

```
/childroot/jailid/user/docs
```

Within this path, each document gets its own sub-directory based on another random Child-ID (which could be the Process ID of the Kit). This ChildId will be given out to clients to facilitate the insertion and downloading of documents. (Although strictly speaking the client can use the main document URI as key, this is the current design.)

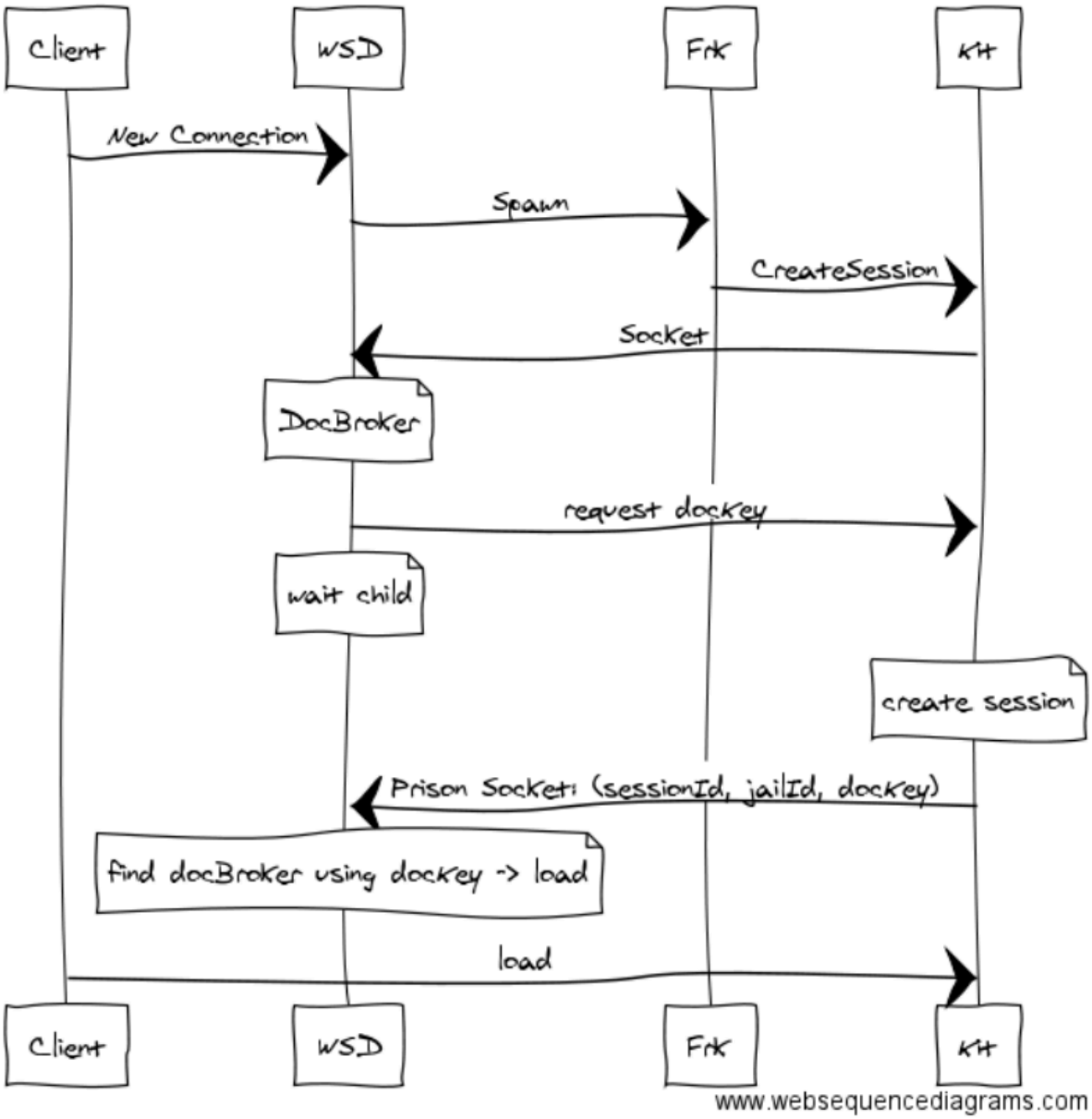
```
/childroot/jailid/user/docs/childid
```

10.7 Client Connection

A request from a client to load a document will trigger the following chain of events.

- WSD public socket will receive the connection request followed by a "load" command. The connection includes the `wopiSrc` unique URL, which includes the user's token.

- An instance of DocumentBroker with the given `wopiSrc` (without the user token) is searched for, if one exists, the document was loaded and this is a new view to the existing document. Otherwise, a new DocumentBroker instance is created and registered internally.
- WSD finds an available Kit process. If none is available, a request is made to ForKit to spawn more.
- A ClientSession (ToClient) is created and takes ownership of the incoming socket to handle the client traffic.
- ForKit sends Kit request to host URI via internal Unix-Socket.
- Kit connects to WSD on an internal port.
- The Kit internally creates Document and ChildSession instances to abstract the document and views on it, respectively.
- WSD creates another ClientSession (ToPrisoner) to service Kit.
- ClientSession (ToClient) is linked to the ToPrisoner instance, copies the document into jail (first load only) and sends (via ToPrisoner) the load request to Kit.
- Kit loads the document and sets up callbacks with LOKit.
- ClientSession (ToClient) and ClientSession (ToPrisoner) tunnel the traffic between clients and the Kit both ways.



10.8 Tile Rendering

The document is rendered into raster images on the server (the Kit) and sent to the client in pre-defined dimensions called tiles. The tiles are tracked on the client and displayed.

When there is a modification (by any other user, or the current one) the Kit will send invalidation notifications to all (active) clients. Each client in its turn will send requests to render the tiles that are out of date.

The server tracks the requests from all clients and renders each unique tile request only once. A tile is not unique only by its coordinates and size, but also by the zoom factor. This makes sure that no tile is rendered more than once. The tiles are cached, so subsequent requests to the same unique tile is served from cache.

Rendering is expensive, and it pays to be minimize them where possible.

10.9 Protocol

The protocol between the client and server uses plain-text with the occasional JSON, if structured data is needed. It is documented separately.

Payloads, in some cases, need to be in binary. This is the case, for example, for rendered tiles. These tile responses must contain the binary data of the tiles they contain.

10.10 Threading

The threading model is as simple as possible. Specifically, each document is handled on the WSD side with a single thread. Similarly, in the Kit, each document has a primary single thread. In both of these cases, this primary thread is responsible for the socket communication as well as the handling of commands/events.

On the WSD side, the DocumentBroker is the owner of this thread. It is regulated through the poll system call, which, when there is no new data in the sockets to read, and no data to write, puts the thread into efficient wait state for new input from any of the sockets (belonging to that particular document), or a timeout. This approach is both simple (minimal thread synchronization concerns) and efficient.

Similarly, in the Kit, the same thread that handles document input, commands, and events, is the same thread that handles the socket logic. The way this is done is through the `runLoop()` Kit API that registers callbacks that the document's main thread calls in its main loop.

10.11 File Server

CoolWSD acts as a file server, in addition to being the server component that handles document loading, editing, saving, etc. The file server in CoolWSD serves only known files. That is, the files are known in advance, enumerated, read from disk, loaded into an in-memory cache, all during starting up and initializing the server.

This has a number of benefits, beyond performance. First, only the known directory (browser/dist) is served from. This avoids the risk of exposing any files outside of the known directory. Second, files that should not be served (for any reason) are detected and excluded at start up. In addition, the file server is responsible for serving service-specific files, such as hosting and discovery, as well as `favicon.ico`, `robots.txt`, etc.

Once the file server is initialized, it can serve only the files that it has cached in memory. All other requests are ignored, with proper error logged and an HTTP error code returned (such as 403 or 404).

File serving is exclusively done over HTTP and HTTPS only. The only HTTP supported verbs are GET and POST. GET is used for file serving while POST is exclusively for interacting with documents. For example, converting, downloading, etc.

The reason for using POST for document interactions is to pass the authentication key, which is used to authenticate the user against the storage server and against accessing the document in question, is safest when transmitted in the body of the request rather than in the address, as GET would do.

Finally, for loading documents and for the connection between the server and client, the HTTP socket is upgraded to WebSocket, which is used for the duration of the session that a user has on a document.

10.12 Communication Security

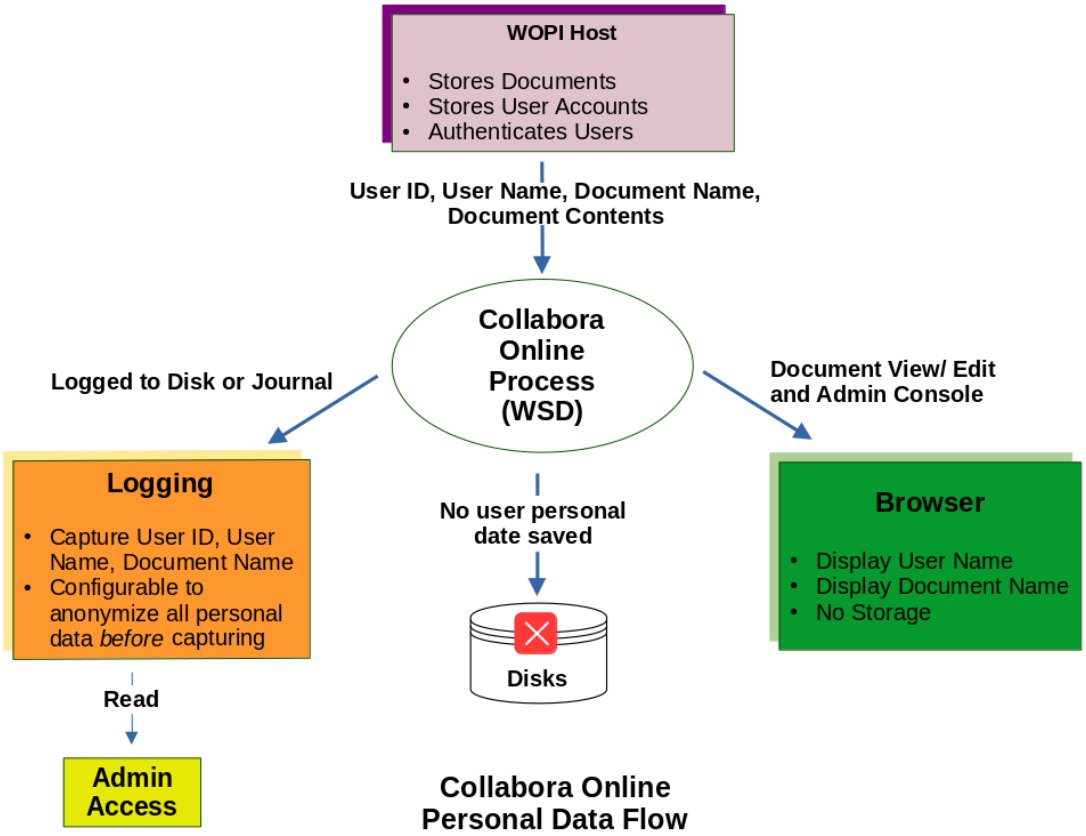
When the host integration client initiates a request to view or edit a document in Collabora Online, it will generate and transmit an Section 5.3 to the WSD. The WSD will send back this authentication token with any WOPI request used to read or save the document. This token will need to be verified by the host (the WOPI Server). The WOPI request is done over HTTPS, and WSD will verify the validity of the TLS certificate of the host when performing WOPI request. This is the access control mechanism for the publicly facing part of Collabora Online.

Behind, the WSD will establish a client connection between itself and a new instance of LOKit that was just created, instance contained in a chroot jail, where system calls and file system access are limited and where only one document is loaded at a time. The LOKit instance and WSD are run the same machine and the communication is done over UNIX domain socket. The legitimacy of the connection is verified using the standard UNIX socket peer credentials mechanism and matching of uid / gid.

Note that this doesn't protect against any tampering done by the super-user (root) on the machine running WSD and LOKit.

Personal data flow

The only place where Collabora Online interacts with user data is what it gets from CheckFileInfo (including the document name). That goes to two places: logs and user interface. The logs can be disabled via the anonymize feature, and the user interface is transient (no storage).



Cookies and local storage

Collabora Online (the online office suite) does not use cookies. However the Section 2.7.10 of Collabora Online uses a session cookie to store a JSON Web Token (JWT) for authentication.

User preferences are stored in browser's local storage. For example:

- dark mode or light mode
- compact view (classic menu + toolbar) or tabbed view (notebookbar)
- whether to show sidebar
- whether to show document navigator
- whether to show status bar
- automatic spell checking on or off
- last used colors in color palettes
- accessibility support on or off (when it's enabled on server's side)
- last used citation style (when Zotero integration is present)

The local storage is also used to store data to control the display of Welcome and User Feedback dialogs in CODE.

Chapter 13

PostMessage API

PostMessage API is used to interact with parent frame when Collabora Online's browser part is enclosed in one. This is useful for hosts wanting to integrate Collabora Online in them.

This API is mostly based on WOPI specification with few extensions/modifications. All messages sent are in this form :

```
{
    "MessageId": "<MessageId>",
    "SendTime": "<Timestamp when message is sent>",
    "Values": {
        "<key>": "<value>"
    }
}
```

`SendTime` is the timestamp returned by browsers' Date.now(). The post messages sent from the WOPI host should also be in same form.

It is to be noted that as mentioned in WOPI specs, Collabora Online frame will ignore all post messages coming from the host frame if Host_PostmessageReady has not been received. Further, since for embedding Collabora Online as an iframe WOPI implementation is a must, it is required that `PostMessageOrigin` property is present in WOPI host's CheckFileInfo response. Otherwise, no post messages will be emitted.

13.1 Initialization

Table 13.1. Editor to WOPI host

MessageId	Values	Description
App_LoadingStatus	Status: **<String>** DocumentLoadedTime: **<Timestamp>**	If status is Frame_Ready, Collabora Online frame is loaded and UI can be shown. Accompanying keys: Features: This client's capabilities. Supported values are: VersionStates. Tells the host that client supports different version states. See Version Restore for more details. When Status is Document_Loaded, document has been completely loaded and host can also start using PostMessage API. Accompanying keys: DocumentLoadedTime. When Status is Failed, document hasn't been loaded but host can show the Collabora Online frame to present an error for a user. When Status is Initialized, postMessage listener in editor has been initialized. WOPI host can start to send postMessages. It is used to send early postMessages even before Frame_Ready.

Table 13.2. WOPI host to editor

MessageId	Values	Description
Host_PostmessageReady		See WOPI docs for detail.

13.2 Query

You can query data from the editor using post message API. All responses are returned with query's MessageId suffixed with '_Resp' as shown below Getters

WOPI Host to Editor

MessageId	Values	Description
Get_Views		Queries the editor for currently active views of the document. Response is returned in form of Get_Views_Resp
Get_Export_Formats		Queries the editor for all the supported export formats for currently opened document. Response is returned in form of Get_Export_Formats_Resp
Get_User_State		Queries the editor for current user activity state (is idle or active). Response is returned in form of Get_User_State_Resp

Getters response

Editor to WOPI host

MessageId	Values	Description
Get_Views_Resp	ViewId: **<Number>** UserId: **<String>** UserName: **<String>** Color: **<Number>** ReadOnly: **<Boolean>** IsCurrentView: **<Boolean>**	Give details of all current views when queried using Get_Views
Get_Export_Formats_Resp	Label: **<String>** Format: **<String>**	Response to query Get_Export_Formats. Label would contain a localized string explaining about the format. Format is the file extension of the format which is required while requesting export of the document.
Get_User_State_Resp	State: **<String>** Elapsed: **<Number>**	Response to query Get_User_State. State would contain a non-localized string with the activity state ("active" or "idle"). Elapsed is the number of seconds since the last activity of the user.

13.3 Session Management

13.3.1 WOPI Host to editor

MessageId	Values	Description
Action_RemoveView	ViewId: <Number>	Remove the session.

13.3.2 Editor to WOPI Host

MessageId	Values	Description
View_Added	ViewId: **<Number>** UserId: **<String>** UserName: **<String>** Color: **<Number>** ReadOnly:**<Boolean>** Deprecated: true;	A new member is added. ViewId is unique integer identifying a session/view. UserId is user identity. UserName is display name of the user. Color is RGB color integer value. ReadOnly tells if the new view is opened as read-only. This message is **deprecated**, opened as read-only. This message is **deprecated**, instead implement just handling of Views_List which holds the same payload as Get_Views_Resp.

View_Removed	ViewId: <Number> Deprecated: true;	View with ViewId has closed the document. This message is **deprecated**, instead implement just handling of Get_Views_Resp and if you need the info which view has been added / removed, check against the previous state. This message is **deprecated**, instead implement just handling of Views_List which holds the same payload as Get_Views_Resp.
Views_List	See Get_Views_Resp.	Complete information about the currently connected views.
User_Idle		Indicates that user become idle. It happens after longer inactivity time defined in the configuration file /etc/coolwsd/coolwsd.xml
User_Active		Indicates that user become active again. It happens after user clicked on the idle screen.

13.4 Actions

13.4.1 WOPI host to editor

MessageId	Values	Description
Action_Save	DontTerminateEdit: <boolean> DontSaveIfUnmodified: <boolean> Notify: <boolean> ExtendedData: <String>	Saves the document. DontTerminateEdit is relevant for spreadsheets where saving a document can terminate the edit mode (text cursor disappearing). Setting this to true won't terminate the edit mode and can be used to save document without disrupting user's editing session in spreadsheets. DontSaveIfUnmodified prevents coolwsd to save the file back to storage if document is unmodified (only cursor position changed etc.) but still saved. This can be helpful to prevent creating unnecessary file revisions. Notify when present and set to true notifies the host when document is saved. See Action_Save_Resp for details. ExtendedData optional data carried over to the WOPI host if provided in the X-COOL-WOPI-ExtendedData header. The contents are preserved as-is, however, care must be taken to avoid using anything that HTTP headers do not allow, also, special values such as new-line, null character, non-printable characters, etc. are not allowed. The client can use this to pass multiple values to the WOPI host which can then act on them.
Action_SaveAs	Filename: <String> Notify: <boolean>	Creates copy of the document with given Filename. Filename is the requested filename for the new file. Notify when present and set to true notifies the host when document is saved. See Action_Save_Resp for details.

Action_FollowUser	Follow: **<Boolean>** ViewId: **<Number>**	Turn on or off the follow user feature. When Follow is set to true or is not define set to true or is not enables following the editor, disables following when set to false When Follow is set to true or is not defined ViewId parameter specifies user to follow. When ViewId is not defined, the current editor is followed.
Action_Close		Closes the document.
Action_Fullscreen		Switch to fullscreen mode.
Action_FullscreenPresentation	StartSlideNumber: **<Number>** CurrentSlide: **<Boolean>**	Start the presentation in Impress. StartSlideNumber optionally specify the starting slide (from 0) If CurrentSlide is true, then the presentation starts from the current slide.
Action_Print		Prints the document.
Action_Export	Format: **<String>** Notify: **<boolean>**	Downloads the document in format specified by Format. Format must be from the list returned in Get_Export_Formats. Notify when present and set to true notifies the host when document is saved. See Action_Save_Resp for details.
Action_InsertGraphics	url: **<String>**	Downloads image from the url and inserts it to the document. This is usually the response to a successful UI_InsertGraphic
Action_ShowBusy	Label: **<String>**	Shows an in-progress overlay, just like what appears when saving the document, with the given Label.
Action_HideBusy		Hides any in-progress overlay, if present.
Action_ChangeUIMode	Mode: 'classic' | 'notebookbar'	Changes the user interface: Classic Toolbar or Notebookbar.
Action_Paste	Mimetype: **<string>** Data: **<string>**	Pastes the data directly to the document bypassing the internal paste mechanism. Example: Values: {Mimetype: "text/plain;charset=utf-8", Data: "foo"}};

13.4.2 WOPI editor to host (Response)

MessageId	Values	Description

`Action_Load_Resp`	`success: <boolean>`
	`result: <string>`
	`errorMsg: <string>`
	`errorType: <string>`
		Acknowledgment when load finishes. `success` tells if COOL was able to load the document successfully.
		`result` contains the reason the document was not loaded.
		`errorMsg` contains a detailed error message in case loading failed. Probably it will contain the error message returned from the WOPI host.
		`errorType` contains optional error identifier based on which WOPI host can customize error message.

`Action_Save_Resp`	`success: <boolean>`
	`result: <string>`
	`errorMsg: <string>`
	`fileName: <string>`
		Acknowledgment when save finishes. This response is only emitted if `Notify` parameter is mentioned by Action_Save PostMessage API.
		`success` tells if COOL was able to save the document successfully.
		`result` contains the reason the document was not saved. In case, document was not saved because it was not modified, then this parameter contains the string 'unmodified'. In this case, WOPI hosts can be sure that there are no changes pending in the document to be saved to the storage.
		`errorMsg` contains a detailed error message in case saving failed. Probably it will contain the error message returned from the WOPI host.
		`fileName` if success equals true then contains saved file name.

FollowUser_Changed	FollowedViewId: <Number> IsFollowUser: <Boolean> IsFollowEditor: <Boolean>	Notification about current following state. FollowedViewId tells which user is followed. IsFollowUser determines if following the specific user is activated. IsFollowEditor determines if following the editor is activated. If both IsFollowUser and IsFollowEditor are false then following is inactive.
Action_ChangeUIMode_Resp	Mode: <string>	Notification about UI mode switch (Tabbed/Compact) Mode tells which mode will be used.

13.5 Version Restore

13.5.1 WOPI host to editor

MessageId	Values	Description
Host_VersionRestore	Status: <string>	The Only possible value is Pre_Restore. This message is sent by the host **before** actually restoring the document and **after** user showed the intent to restore the document. This is so such that if there are any unsaved changes, Online can save them to storage before document is restored.

13.5.2 Editor to WOPI host

MessageId	Values	Description
App_VersionRestore	Status: <string>	This is the reply for the Host_VersionRestore message. Possible values for Status (for now) is: Pre_Restore_Ack. It means that host can go ahead with restoring the document to an earlier revision.

Note

These messages are only emitted if *App_LoadingStatus* contains *VersionStates* in *Features*. Otherwise, host can immediately restore the version to earlier revision.

13.6 Miscellaneous

13.6.1 WOPI host to editor

MessageId	Values	Description
Insert_Button	id: **<string>** imgurl: **<string>** hint: **<string>** accessKey: **<string>** mobile: **<boolean>** tablet: **<boolean>** label: **<string>** insertBefore: **<string>** unoCommand: **<string>**	Inserts a button to the top toolbar. It responds with Clicked_Button post message event on which hosts can react accordingly (except when the unoCommand id parameter is a unique id of the toolbar button. It is recommended to prefix such ids given here given here with some host namespace so that it doesn't conflict with existing toolbar IDs. In case of conflict, button is not added. imgurl parameter is the link to the image that will be set as button image in the toolbar. The ideal size of the image is 24x24px. The image must be hosted on the host URL to not violate Content-Security-Policy. hint This is used as a tooltip of the button. This is used by accelerator definitions class. Users can press alt or alt+shift and use this key. One should be careful to not introduce conflicting access keys. mobile whether the button should be shown when the interface switches to mobile mode. tablet whether the button should be shown in tablet mode (true if absent) label When a read-only document is opened, we don't have the toolbar at all. In this case, this newly added button is present in file menubar. The text against this label is used as text of the menubar item. insertBefore Specify the position where the button should be inserted. insertBefore is the button ID (see Finding toolbar button IDs). unoCommand UNO Command to be executed on button click (Reference). When this property is set, no 'Clicked_Button' response is sent. The button click will be handled by LibreOffice.
Hide_Button	id: **<string>**	Hides a button from the toolbar. id is the button ID (see Finding toolbar button IDs).
Show_Button	id: **<string>**	Hides a button from the toolbar. id is the button ID (see Finding toolbar button IDs).
Remove_Button	id: **<string>**	Removes a button from the toolbar. id is the button ID (see Finding toolbar button IDs).

`Hide_Command`	`id: <string>`	Hide the UI for a command. This include the menu items and toolbar buttons. `id` is the command (see Finding toolbar button IDs).
`Show_Command`	`id: <string>`	Show the UI for a command. The reverse of `Hide_Command`. `id` is the command (see Finding toolbar button IDs).
`Collapse_Notebookbar`		Hide the notebook bar buttons. This is equivalent to double clicking on the name. Click on the name will show it again. This also hide the button strip in the classic UI.
`Extend_Notebookbar`		Show the notebook bar buttons. This is the equivalent of clicking on the name. This also show the button strip in classic UI.
`Hide_StatusBar`		Hides the status bar
`Show_StatusBar`		Shows the status bar
`Remove_Statusbar_Element`	`id: <string>`	Removes an element from the status bar. `id` is the element ID (see Finding status bar element IDs).
`Hide_Menubar`		Hides the menu bar.
`Show_Menubar`		Shows the menu bar.
`Grab_Focus`		This restores focus to the application, activating it, and removing any overlay indicating quiescence, and re-connecting to the server if necessary. Useful after leaving the application for a lengthy period, or when wanting to restore browser focus after presenting an overlaid dialog.
`Hide_Ruler`		Hides the horizontal document ruler (Writer only)
`Show_Ruler`		Shows the horizontal document ruler (Writer only)
`Hide_Menu_Item`	`id: <string>`	Hides an item from the menu. `id` is the item ID as defined in the browser/src/control/Control.Menubar.js.
`Show_Menu_Item`	`id: <string>`	Shows an item from the menu. *id`* is the item ID as defined in the browser/src/control/Control.Menubar.js.

Disable_Default_UIAction	action: **<string>** disable: **<Boolean>**	Disable the default handler and action for a UI command. action is the action name to enable/disable the default action for. disable controls whether to disable (true) or enable (false) the default action. When set to true, the given UI command will only issue a postMessage without invoking the default action, leaving it up to the client to intercept the postMessage event and handle as necessary. Notice that some actions do not have any default handler to begin with (such as UI_SaveAs and UI_Share) and therefore this will have no effect on them; they only issue postMessage notification anyway without taking any action beyond that. For example, UI_Save will be issued for invoking the save command (from the menu, toolbar, or keyboard shortcut) and no action will take place if UI_Save is disabled via the Disable_Default_UIAction command. Clients who disable UI_Save should then issue Action_Save themselves, when and if they desire to save the document. Similarly, when disabling UI_Close, the document will not close upon invoking the UI_Close action, instead a postMessage notification will be issued and it will be up to the client to issue Action_Close when they desire. Clients must be careful not to issue duplicate actions when the default handler is enabled, instead, they should only issue actions themselves when the default is disabled. Note: currently only UI_Save and UI_Close are supported.
Send_UNO_Command	Command: **<string>** Args: **<object>**	Send an UNO command to the editor. See examples in browser/html/framed.doc.html.
Error_Messages	list: [{ type: **<identifier>**, msg: **<string>** }, { type: **<identifier>**, msg: **<string>** }.....]	Send list to override error messages available in the editor. type identifies which message to override For example to override loadfailed: `list: [{ type: 'loadfailed', msg: 'your custom msg' }]` To get list of error messages available to override, please checkout browser/src/errormessages.js.

`Hint_OnscreenKeyboard`		Indicate that the device uses on screen keyboard (like a tablet). This is useful if the detection failed to identify the device is a tablet. This is useful if the device environment is unusual, and should only be used with great caution.
`Hint_NoOnscreenKeyboard`		Indicate that the device is not a tablet. This is the opposite of `Hint_OnscreenKeyboard`

13.6.2 Finding toolbar button IDs

Toolbar button IDs are defined in `getToolItems/create` functions in:

- Control.TopToolbar.js for the top toolbar on desktop or tablet.
- Control.MobileTopBar.js for the top toolbar on smartphone.
- Control.MobileBottomBar.js for the bottom toolbar on smartphone.
- Control.StatusBar.js for the statusbar on desktop.

Note that they usually don't change but there is no guarantee that they are stable.

13.6.3 Finding status bar element IDs

Status bar button IDs are defined in the `onDocLayerInit` function in Control.StatusBar.js. Note that they usually don't change but there is no guarantee that they are stable.

13.6.4 Editor to WOPI host

MessageId	Values	Description
`Clicked_Button`	`id: <string>`	This event is emitted when the custom button added via Insert_Button is clicked.
`Download_As`	`Type: 'print'|'slideshow'|'export'` `URL: <string>`	This event is emitted when the user chooses 'Print' or 'Show slideshow' or 'Download As `[some type']` and the integration indicates via `DownloadAsPostMessage` in the `CheckFileInfo` that it wants to handle the downloading of pdf for printing or svg for slideshows or exported document. This is in situations when the integration cannot rely on browser's support for downloading like in mobile apps that use the Online in a Web View.
`UI_OpenDocument`		Requests WOPI host to open a popup window where user can pick another document to view & edit.

`UI_CreateFile`		Requests WOPI host to open a new browser tab and create a new document. The document type is passed as `DocumentType` argument, and can be 'text','spreadsheet','presentation' or 'drawing'.
`UI_SaveAs`	`Args: {format: '<extension>' }`	Requests WOPI host to create appropriate UI, so that the user can choose path and File name for creating a copy of the current file. Response to this query is sent via `Action_SaveAs` message. Optional arguments: file extension. When this parameter is passed a dropdown appears and the newly saved file is loaded in the integration instead of downloaded.
`UI_InsertGraphic`		Requests WOPI host to open a popup window where user can pick an image to insert. A successful pick would lead to the WOPI host sending a `Action_InsertGraphics` postMessage back.
`UI_Cancel_Password`		Notifies WOPI host that the user clicked on the 'cancel' option when opening a password protected file, instead of providing the password to decrypt it.
`UI_Hyperlink`		Notifies WOPI host that the user clicked a hyperlink and confirmed they really want to leave the document to follow the hyperlink. This is especially useful for integrations that embed Collabora Online into an `iframe` in a mobile app, where actually trying to open a new window should trigger starting a new Activity on Android (or something similar on iOS). The integration using this most probably also wants to trigger the `Disable_Default_UIAction` for `UI_Hyperlink`.

Doc_ModifiedStatus		Notification to update the modified status of the document. Values.Modified will be true, if the document has been modified since the last save, otherwise, it will be false if the document has been saved. Note that this notification may be published without a change from the prior value, so care must be taken to check the Values.Modified value and not assume the notification itself implies the modified state of the document on its own.

13.7 Calling Python scripts

13.7.1 WOPI host to editor

MessageId	Values	Description
CallPythonScript	script. The Values ScriptFile: **<string>** Function: **<string>** Values: **<object>**	Calls a Python parameter contains an object with named parameters that are passed to the script.

13.7.2 Editor to WOPI host

MessageId	Values	Description
CallPythonScript-Result	commandName: **<string>** Values: **<object>**	Returns the result The URL of the script called is in the commandName parameter.

13.8 Mentions

Note Mentions are only working in Writer for now

13.8.1 Editor to WOPI host

MessageId	Values	Description
UI_Mention	type: autocomplete text: **<string>**	When user starts typing with "@", CollaboraOnline will send this postMessage with partial text followed by "@" to get the list of usernames from the integrator
	type: selected username: **<string>**	When user selects the a username from list given by the integrator this message gets fired

13.8.2 WOPI host to editor

MessageId	Values	Description

Action_Mention	list: [{
	username:
	"example-username1",
	profile:
	"link-to-the-profile"
	},
	{
	username:
	"example-username2",
	profile:
	"link-to-the-profile"
	}.....]
		Based on UI_Mention message of type autocomplete, integrator should send this message with list of user object. Each user object contains username and profile

Conversion API

Collabora Online allows you to convert between various file formats easily. To do so, all you need to do is to HTTP POST the content of the file to the specific endpoint.

API: HTTP POST to `/cool/convert-to/<format>&<lang=xx-XX>`

- the format is e.g. `png`, `pdf` or `txt`
- the pdf version (optional) for the respective type of PDF to be used for the output file. Example: `PDFVer=PDF/A-2b`
- the file itself in the payload.
- the language parameter (optional) sets the default format language, useful for date type cells. If passed, the load language is used and it determines the display/output format. Example: `lang=fr-FR`

Example:

```
curl -k -F "data=@test.txt" https://localhost:9980/cool/convert-to/docx > out.docx
```

Note: The `-k` in this example disables certificate validation as it is unlikely you have one in that situation. **On a production system, please make sure you have valid certificates and to not disable the validation with** `-k`.

- or in HTML:

```
<form action="https://localhost:9980/cool/convert-to/docx"
enctype="multipart/form-data" method="post">
 File: <input type="file" name="data"><br/>
 <input type="submit" value="Convert to DOCX">
</form>
```

Alternatively you can omit the `<format>`, and instead provide it as another parameter.

Example:

```
curl -k -F "data=@test.odt" -F "format=pdf" -F "PDFVer=PDF/A-2b"
https://localhost:9980/cool/convert-to > out.pdf
# PDFVer is optional, PDF versions currently supported:
# PDF/A-1b, PDF/A-2b, PDF/A-3b, PDF-1.5, PDF-1.6
```

Note: Same as above, the `-k` in the example disables certificate validation.

- or in HTML:

```
<form action="https://localhost:9980/cool/convert-to"
enctype="multipart/form-data" method="post">
```

```
 File: <input type="file" name="data"><br/>
 Format: <input type="text" name="format"><br/>
 <input type="submit" value="Convert">
</form>
```

Note: the `convert-to` endpoint is restricted to allowed host addresses that can be set in the `/etc/coolwsd/coolwsd.xml` configuration file. The IP addresses have to be added as dot-escaped `net.post_allow.host` entries.

Linking API

Collabora Online allows you to extract list of objects inside document you can link to. Received targets can be then used to open document at specific position or generate it's thumbnail.

API: HTTP POST to `/cool/extract-link-targets`

- the file itself in the payload.

Example:

```
curl -F "data=@file.docx" https://localhost:9980/cool/extract-link-targets >
targets.json
```

Example output:

```
{
    "Targets": {
        "Tables": {
            "Table1": "Table1|table"
        },
        "Frames": {},
        "Images": {
            "image7.png": "image7.png|graphic"
        },
        "OLE objects": {},
        "Sections": {
            "Table of Contents1": "Table of Contents1|region"
        },
        "Headings": {},
        "Bookmarks": {
            "_lh2zfxamp5al": "_lh2zfxamp5al"
        },
        "Drawing objects": {}
    }
}
```

Note: You can open document at specific target by using additional URL parameter for example: `&target=Table1|table`

API: HTTP POST to `/cool/get-thumbnail`

- the file itself in the payload.
- optional: target parameter for which we want to generate thumbnail `Table1|table`

Example:

```
curl -F "data=@file.docx" -F "target=Table1|table"
https://localhost:9980/cool/get-thumbnail > thumb.png
```

Note: the endpoints are restricted to allowed host addresses that can be set in the `/etc/coolwsd/coolwsd.xml` configuration file. The IP addresses have to be added as dot-escaped `net.post_allow.host` entries.

Using Python scripting in Collabora Online

This document describes the Python scripting feature to manipulate documents being edited in Collabora Online.

16.1 Description

Python scripting allows for document manipulation while being edited in Collabora Online. The use-case assumes a web page has a Collabora Online document instance open in an iframe and that the functionality that wants to access the Python scripting is on the parts of the web page outside that iframe.

16.2 Background

The core LibreOffice has support for Python scripting already. Enhancements are made so that a Python script (or other code in the LibreOffice core) can return structured values to the JavaScript code in the browser, for instance lists, and not just simple strings or integers.

On the Online side, notifications from returning Python scripts are forwarded to the JavaScript code running in the browser.

16.3 Python script files

The Python script files containing functions to be called should be located in the LibreOffice installation, in the folder `share/Scripts/python`. (In the LibreOffice sources, they are in `scripting/examples/python`.) Currently, this folder contains `Capitalise.py`, `HelloWorld.py`, `InsertText.py`, `NamedRanges.py`, and `SetSellColor.py` and files that are used by the demo web page `framed.html` (see below), and a few others. Customer-specific Python files should be placed in the same location. After editing script files, the Collabora Online instance must be restarted.

These server-side scripts are read-only during run-time and cannot be modified via Online. Furthermore, the execution of Python scripting is limited to the Python script files in the aforementioned directory, which are prepared in advance. Arbitrary Python code execution is not possible, nor is the execution of dynamically code generated code. These are to guarantee strong security constraints.

Additional security is provided by not enabling the Python script provider by default. To enable it, you need to explicitly install, from the respective customer or CODE repositories, the following packages:

- `collaboraofficebasis-python-script-provider`, makes it possible to implement `uno` “scripts” in python,

- `collaboraofficebasis-pyuno`, makes it possible to implement `uno` components in python.

16.4 Instructions

In the Online sources there is a web page `browser/html/framed.html` and `browser/html/framed.doc.html` that are examples of web pages that run an unmodified Collabora Online instance inside an HTML `iframe`, and then from the HTML code outside the iframe calls Python scripts in the underlying LibreOffice instance to manipulate data in the document open in the Collabora Online instance. Various parameters can be passed to the Python scripts, and return values handled.

Both `framed.html` and `framed.doc.html` are just for demonstration purposes and very bare-bones visually.

16.4.1 Forms and JavaScript in framed.html and framed.doc.html

`framed.html` and `framed.doc.html` contain a set of small HTML forms and corresponding JavaScript functions that are invoked when a form is submitted. When a form is submitted, in this demonstration case, a corresponding JavaScript function is called. Of course in a real-life use case this could be constructed differently. The JavaScript function fetches the input fields and passes them to the JavaScript functionality of the Collabora Online running inside the iframe using the `postMessage()` standard JavaScript function.

The parameter to `postMessage` is a stringified JSON object. The interesting fields in that are: `MessageId`, which should be `CallPythonScript`, `ScriptFile`, which should be the file name of the Python source file containing the Python function to be called, `Function` which should be the name of the Python function to be called, and `Values` which should be a JSON object containing the (named) parameters to that Python function.

The JavaScript function `receiveMessage()` is set up to handle postMessage() events posted to the HTML page from the iframe, and handle especially those corresponding to return values from called Python functions. Those are distinguished by having a `MessageId` of `CallPythonScript-Result`.

Once `receiveMessage()` knows it is handling a return value from a Python script, it checks the `commandName` field which contains the LibreOffice `vnd.sun.star.script` URL of the called Python function. In the `demonstration framed.html` it is `receiveMessage` that then directly does what is necessary depending on the function called. In a more real-life use case, this could be done in some more generic and complex manner of course.

16.4.2 Sample Scripts

These scripts are provided as samples and as starting points for experimentation and further development. Users are encouraged to make copies of them and modify as necessary. Note that they may get overwritten when upgrading the Online packages, so making separate copies is highly recommended to avoid losing any changes.

SetCellColor:

The first form sets the colour of a cell in the (Calc) document open in the Collabora Online instance. The (zero-based) x and y coordinates of the cell, and the colour (in HTML format, like #A0FFA0 for a very light green) are input fields of the form. In this case the Python file is called `SetCellColor.py`, the function is called `SetCellColor`, and the parameters are the x and y coordinates and the colour.

GetNamedRanges:

The second form has no input fields and causes the function GetNamedRanges() in the Python file `NamedRanges.py` to be called. That function takes no parameter but returns a value that is a list of named ranges in the document. The `receiveMessage()` function inserts these into a textarea element.

AddNamedRange:

The third form is used to add a named range to the document. The input fields are as expected, the name and the range. The called Python function is `AddNamedRange`, also in `NamedRanges.py`. The JavaScript to call this is somewhat complicated because of different syntax used in the input fields and parameters passed to that Python function. One could as well put the parameter mangling into Python code, of course.

DeleteNamedRange:

Finally, and simplest sample, is a form to delete a named range.

InsertText:

This script demonstrates inserting a custom text into a Writer document, replacing any selection (if exists), otherwise inserting at the cursor's position.

LanguageTool

Starting with version CODE 22.05, it's possible to enable support for external grammar checking using LanguageTool. New settings are available both on online and on the core side.

17.1 Collabora Online

You can find the respective option within your *coolwsd.xml* where you can set the LanguageTool.org API settings you need within the *languagetool* block. To turn it on, please set "enabled" property to true. The base URL may be *https://api.languagetoolplus.com/v2* if the cloud version is used. However, your data in the document e.g. the text part of it will be sent to the cloud API. Please read the privacy policy: https://languagetool.org/legal/privacy.

Listing 17.1. Languagetool block of coolwsd.xml

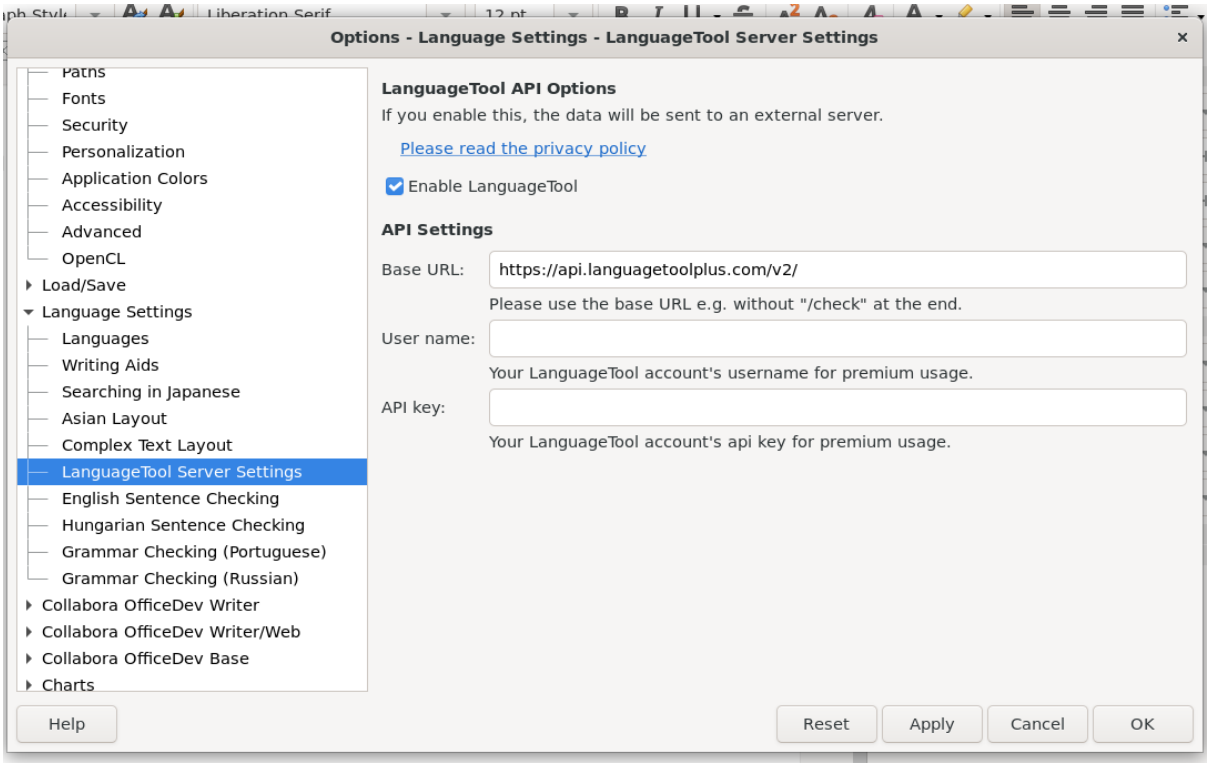
```
 <languagetool desc="LanguageTool Remote API settings for grammar checking">
     <enabled desc="Enable LanguageTool Remote Grammar Checker" type="bool"
default="false">true</enabled>
     <base_url desc="Http endpoint for the LanguageTool API server, without /check
 or /languages postfix at the end." type="string"
default="">https://api.languagetoolplus.com/v2</base_url>
     <user_name desc="LangueTool account username for premium usage."
type="string" default=""></user_name>
     <api_key desc="Api key provided by LanguageTool account for premium usage."
type="string" default=""></api_key>
</languagetool>
```

Please note, *LanguageTool plugin uses `libcurl* and thus the *base_url* has to have *http://* or *https://* as the prefix. Currently, the expected URL scheme should include the API version (as stated in LanguageTool API). E.g.: *https://selfhostedlg.com/v2*

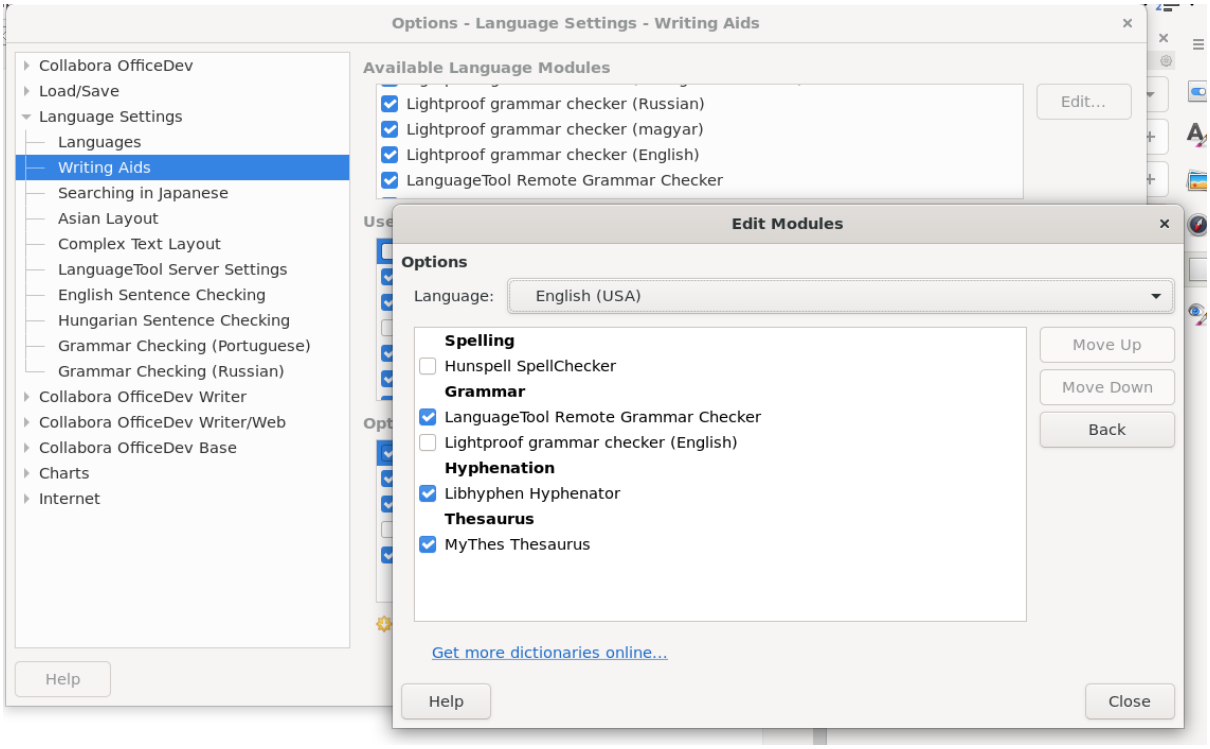
Free and paid versions (including on-premise premium version) are available from LanguageTool.

17.2 LibreOffice Core

A new *LanguageTool Server Settings* group is available in *Options* -> *Language Settings* dialog. Base URL, username and API key can be set here.



Lastly a new *Writing Aid* by the name of *LanguageTool Remote Grammar Checker* should be switched on.



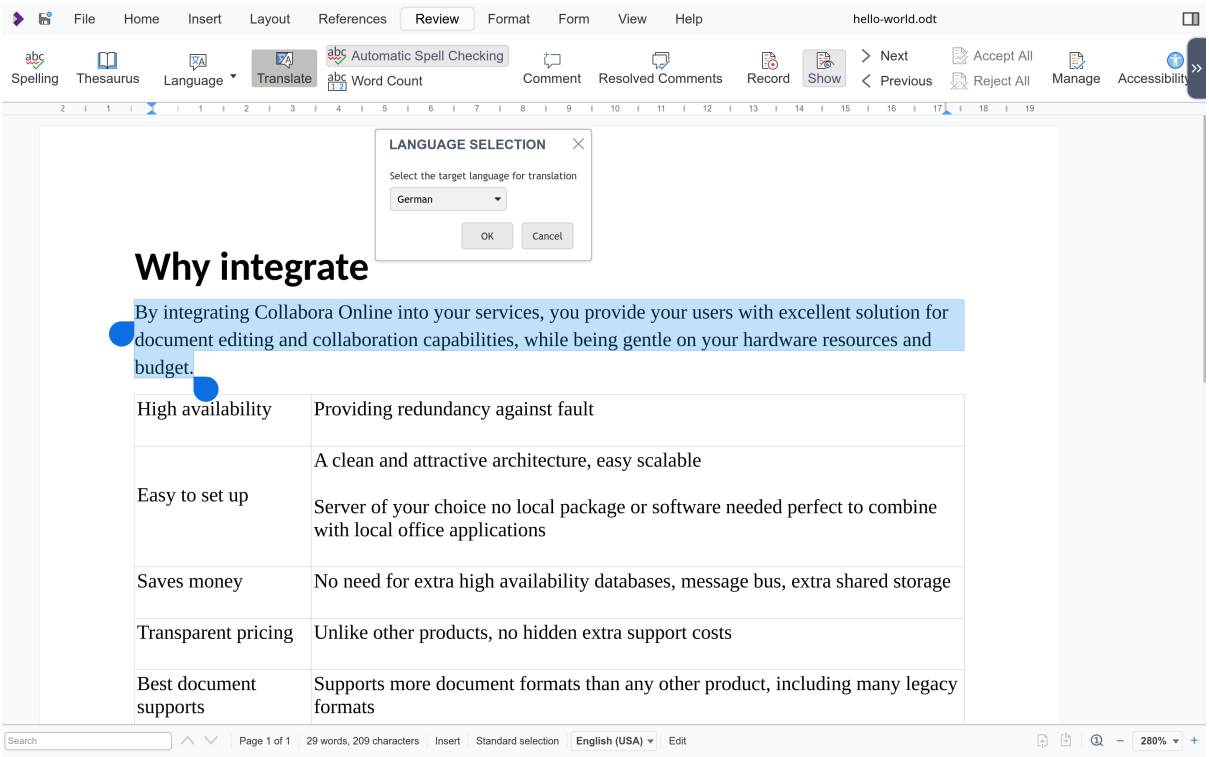
Translator (DeepL)

DeepL Translator is a neural machine translation service that provides an API with programmatic access to DeepL's machine translation technology, making it possible to bring translation capabilities directly to websites and applications. Visit the DeepL API documentation for further information.

18.1 Collabora Online

DeepL is supported by Collabora Online since version 22.05.7.3.

Add your DeepL API URL within `api_url` and your DeepL API key within `auth_key` tags in your configuration (`coolwsd.xml`) to use DeepL's translation capabilities in Collabora Online. For example the selected paragraph can be translated from any supported source language to any supported target language with the Translate button on the Review bar.



You can find the respective option within your `coolwsd.xml` where you can set the DeepL API settings you need within the `deepl` block.

To turn it on, set `enabled` property to true. The API URL may be for example

`https://api-free.deepl.com/v2/translate`, with the free plan. Note that the text content of the document will be sent to the cloud API. Please read DeepL's privacy policy: https://www.deepl.-com/en/privacy .

Listing 18.1. deepl block of coolwsd.xml

```
 <deepl desc="DeepL API settings for translation service">
     <enabled desc="If true, shows translate option as a menu entry in the compact
 view and as an icon in the tabbed view." type="bool"
default="false">true</enabled>
     <api_url desc="URL for the API" type="string"
default="">https://api-free.deepl.com/v2/translate</api_url>
     <auth_key desc="Auth Key generated by your account" type="string"
default=""></auth_key>
 </deepl>
```

Theming of Collabora Online

19.1 How that works and how it looks

Customize it and make it feel at home with your integration. In Collabora Online some parts of the Section 19.2 can be hidden or shown. If you are a Collabora partner or a customer running your own installation, you can also change the theming of Collabora Online. And it can be done very easily too; just by setting a couple of Section 19.4 through your integration. Here's how that works and how it looks.

19.2 User Interface modifications

Some parts of the user interface can be hidden or shown based or what the integration needs. This is controlled by:

```
<input name="ui_defaults" value="VALUES" type="hidden"/>
```

during sending the form when the iframe is being set up (similarly as the `access_token`). The VALUES can have a form like:

```
UIMode=notebookbar;TextRuler=false;PresentationStatusbar=false;SpreadsheetSidebar=false
```

With Collabora Online 21.11 use of `notebookbar` has been deprecated use `tabbed`:

```
UIMode=tabbed;TextRuler=false;PresentationStatusbar=false;SpreadsheetSidebar=false
```

where the:

- `UIMode` specifies the general mode of operator. Possible values are `compact` (`classic` is deprecated) or `tabbed` (`notebookbar` is deprecated).
- `Text`, `Presentation` or `Spreadsheet` - are prefixes to identify the component
- `Ruler`, `Statusbar`, `Sidebar` - are the UI parts that can be affected by this. These are boolean. For example `TextRuler=false` will hide the ruler in Writer.
- `SaveAsMode` when set to `group` will set the layout of the "Save As…" command menu to not be a submenu, but rather list the different format as part of the main menu. Any other value is ignored. By default the different formats are in a submenu from the "Save As…" command.
- `SavedUIState` set to `false` allow forcing the above changes if the user had customised the UI by bypassing the saved state. The use of this option should be limited for case where it's important to always have this default UI.
- `OnscreenKeyboardHint` is tri-state. When set to `true`, assume the device has an onscreen keyboard. When set to `false` assume the device does not have an onscreen keyboard. When

unset, the automatic detection is done. This should only used for specific embedding situation with specific devices.

- `TouchscreenHint` works like `OnscreenKeyboardHint` but force Collabora Online to think it is running on a touch device. This really should only be used as a last resort as well.

The `UIMode` can be also updated after Collabora Online iframe is set up based on a user action through a specific PostMessage call endpoint.

19.3 Extra hidden field in COOL frame integration

In the COOL frame in the integration there is a form where you pass an `access_token` to COOL for loading the valid document. For your theming you have to add another hidden field to the form named `css_variables`. Then the css variables and their values for the theming can be passed, formatted as shown in the example below.

19.4 Content of hidden field "css_variables"

The default values of various css variables can be overridden by sending them in the post message in this format:

```
<input name="css_variables"
value="--co-color-main-text=#000;--co-body-bg=#FFF;--co-txt-accent=#2e1a47;"
type="hidden"/>
```

Note that the variables in the form are formatted slightly different from how they look in the CSS file! The colon `:` found in CSS is replaced by an equal sign `=`.

You can also test your colours by adjusting the COOL url and append those values, so they can be passed via get and so you can see your changes instantaneously. Example:

```
http://localhost:9980/browser/debug.html?file_path=/cool/test/data/hello-world.odt&css_variables
```

19.5 Available variables

Various variables can be overridden for the theming. Their names, and the default values that are used in COOL with the branding package are:

```
--co-primary-element: #4c566a;
--co-primary-element-light: #706aab;
--co-txt-accent: #2e1a47;
--co-primary-text: #ffffff;
--co-border-radius: 3px;
--co-body-bg: #ffffff;
--co-color-main-text: #000000;
```

These are only for the branded versions of COOL. If you are using the CODE version of COOL, you can refer to the color palettes CSS file, regular or dark to find which variables you can override.

19.6 What it is, and how it looks

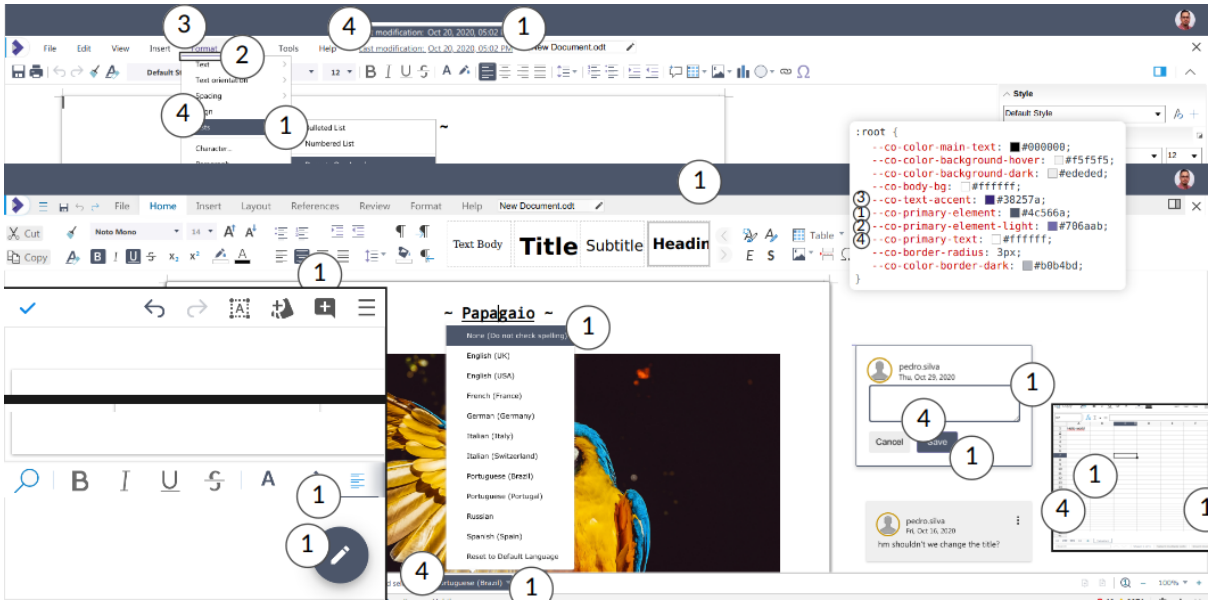


Figure 19.1. New in 6.4: Theme it via CSS Variables

1. `primary-element` is for selected elements on menu's and toolbars, various bars
2. `primary-element-light` is for selected deselected elements
3. `txt-accent`
4. `primary-text` is the text on these elements
5. `border-radius` is the rounding of the selection of items on e.g. toolbars and the status bar
6. `body-bg` is the background beside the document
7. `color-main-text` is the fall-back in the case a specific element does not have its own color text value.

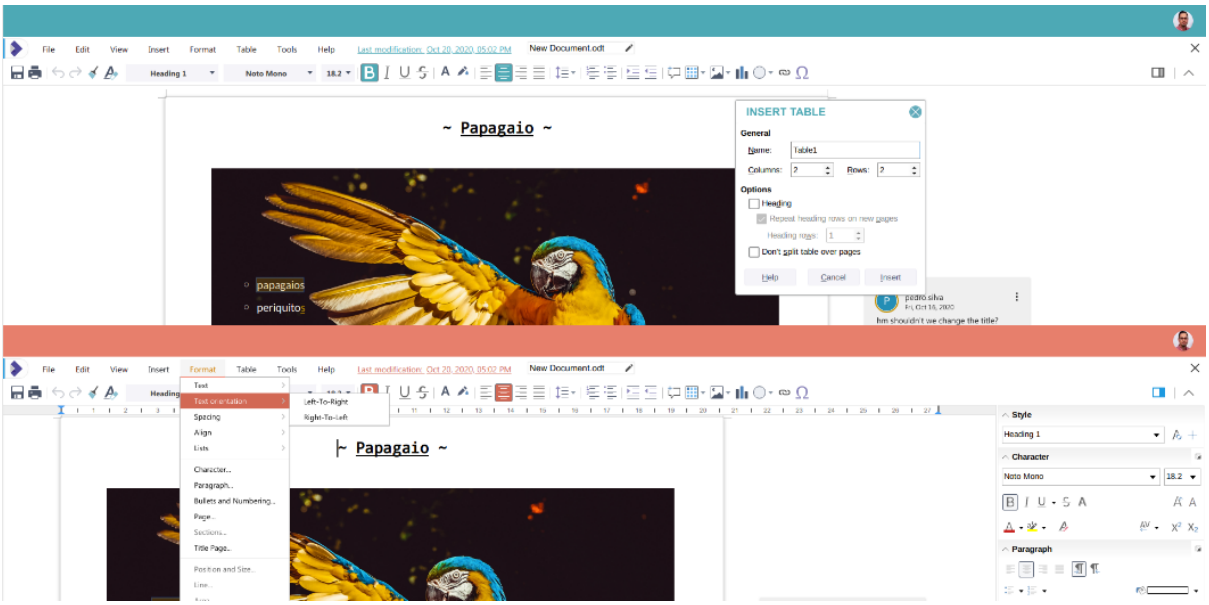


Figure 19.2. Tweak it and make it feel at home with your own integration

Chapter 20

FAQ

20.1 Which programs are included in the office suite?

We include Writer, Calc, Impress and Draw - for text documents, spreadsheets and charts, as well as presentations and drawings.

20.2 Which formats can the programs of Collabora Online read?

All of the key Microsoft file formats both OpenXML and legacy binary file formats - DOC, DOCX, XLS, XLSX, PPT, PPTX as well as RTF and macro enabled versions of these (although we disable macros online by default). Clearly we prefer nicely standardized Open Document formats - ODT, ODS, ODP, ODG. In addition it is possible to import Visio, MS Publisher, WordPerfect files as well as many other flavors. We spend lots of time working on improving our filters and making our interoperability excellent.

20.3 In which formats can documents etc. be exported?

We save back to the core Microsoft file formats, as well as OpenDocument equivalents. We provide PDF download as well.

20.4 Do the programs run in a browser and / or as local clients?

Collabora Online runs on the server in your data center, and is interacted with through a standard browser - there is no need for any client installation. It is necessary however to integrate Collabora Online with another product that will do data storage, and access control for users.

20.5 Do the programs require an uninterrupted network connection?

Our design requires an uninterrupted network connection to get access to your document data. This has many security advantages over more complex DRM based models - potentially allowing an admin to audit and control all access to the document perpetually. It does require a network connection however. If collaboration is not required, then downloading via a sync and share integration (eg. ownCloud, Nextcloud, EGroupware, pydio, Seafile and many others) makes sense - then you can edit off-line and sync when you return.

20.6 Can you add a storage system?

Collabora Online is not tied to any particular storage or authentication system we use a WOPI-like protocol to defer both storage and authentication to the provider we are integrated with.

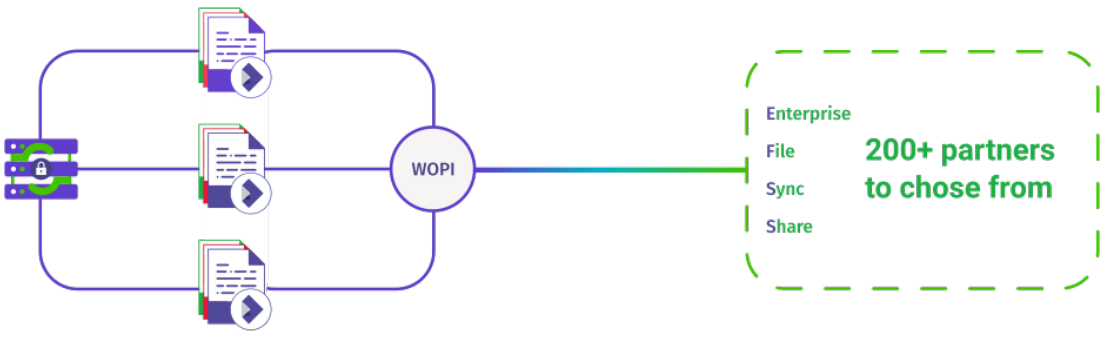
20.7 Can you write and use your own plug-ins with the programs?

Absolutely, we have a huge programming API surface based on the existing LibreOffice UNO APIs. Bespoke uses of Collabora Online can introduce buttons, and rich scripting interactions to communicate from client to server, using the existing standard UNO APIs for LibreOffice.

20.8 High availability, fault tolerance, upgrade-ability?

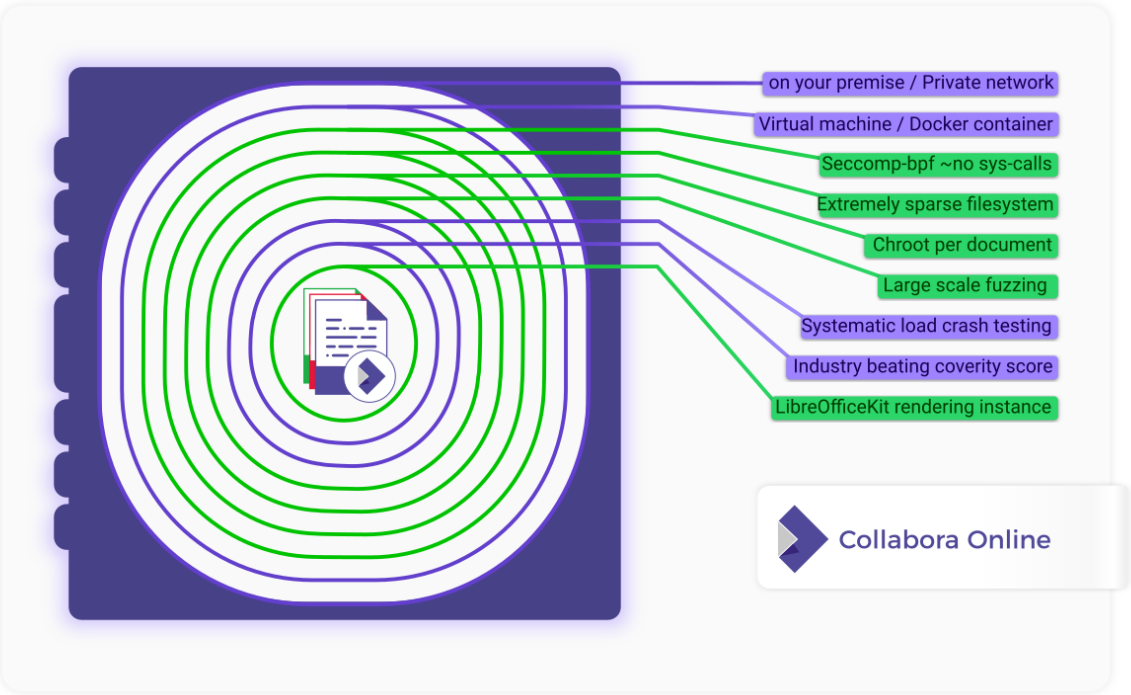
Clearly you want a high availability setup, not only to provide extra scalability, but also to provide redundancy against faults. Collabora Online has a clean and attractive architecture – which scales with your routing network:

- Each document is served by a single node to which all requests and edits are sent for that document by the HA gateway: F5, HA proxy etc..
- Each node is ultimately stateless and needs only limited local storage).
- Collabora Online requires no third party services except of course it needs to connect to your existing file-storage solution.
- Collabora Online regularly saves documents to your existing storage.
- Collabora Online requires only a standard, basic Linux base-system to run on top of.

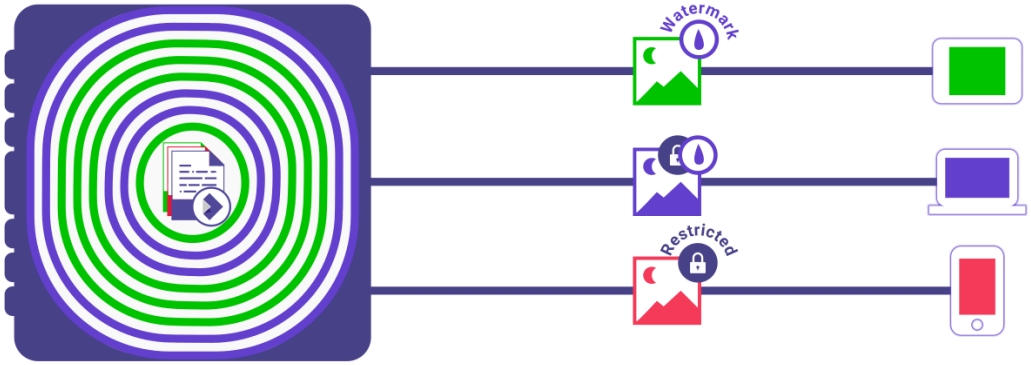


20.9 Do the documents leave the server?

Collabora Online uses an adapted versions of the WOPI standard protocol, and we can use data stores which can provide their own policies. When your document data comes down into Collabora Online we isolate and protect your document in your on-premise server inside a series of concentric security onion shells:



Collabora keeps your document data on the server, and can send only tiled images to the client. These can also be watermarked with the viewer's name. With granular permissions to restrict copy & paste, download, print and so on – Collabora protects your documents like no other.



20.10 What is CODE? Are there non-development editions too?

We offer our free version CODE (Collabora Online Development Edition). CODE is a continuously updated, rolling release where we try out our latest feature work, and has no SLA or long term support. As such, we don't recommend CODE for business or production environments, but you can use it.

If you are familiar with Linux, then CODE would be like our Fedora or openSUSE version - rather than RHEL or SLES, and lots of people use it.

More information here: https://www.collaboraoffice.com/code/

20.11 What are the possibilities to test your products?

For test purpose, we offer our on-line demo (the only solution hosted by us) and CODE (on-premise), our development edition.

Our online demo is the only solution hosted by Collabora, focused on providing to our potential customers and partners an easy way to test Collabora Online with different FSS solutions such as Nextcloud or ownCloud. We don't have a time limit on our demo account, so you can try it out as long as you like. The only limitation we have is space. A demo account has 10MB of space free for testing out your office files. So, it has only test purposes.

https://www.collaboraoffice.com/demo/

We offer our free version CODE. CODE is a continuously updated, rolling release where we try out our latest feature work, and has no SLA or long term support. As such, we don't recommend CODE for business or production environments, but you can use it.

If you are familiar with Linux, then CODE would be like our Fedora or openSUSE version - rather than RHEL or SLES, and lots of people use it.

https://www.collaboraoffice.com/code/

And, finally, we offer Collabora Online supported version as a subscription plan based on the number of users per year with SLA, maintenance, Long term support, software and security updates and much more.

https://www.collaboraoffice.com/collabora-online/

20.12 Can I host your product on my own environment?

We offer an on-premise solution, we don't host the product. Collabora Online is a software component which can be integrated into your web application and enables viewing and editing of, and cooperation on office documents in dozens of file formats. Collabora Online uses unparalleled file format support and rendering capabilities of LibreOffice. The way to integrate into your app is to implement WOPI support, which is very easy and straightforward, and we have the documentation to help you.

It's the very point of Collabora Online to allow editing of documents, and nothing more – the rest, like the document storage, authentication etc. is the responsibility of the those who integrate Collabora Online into their product.

- genindex
- search

- https://github.com/CollaboraOnline/online/