COMPRISE

Cost effective, Multilingual, Privacy-driven voice-enabled Services

www.compriseh2020.eu

Call: H2020-ICT-2018-2020
Topic: ICT-29-2018
Type of action: RIA
Grant agreement Nº: 825081

WP Nº4:	Cost-effective multilingual voice interaction
Deliverable Nº4.5:	Final COMPRISE SDK prototype and documentation
Lead partner:	ASCO
Version Nº:	1.0
Date:	31/05/2021

European Commission

Document information	
Deliverable Nº and title:	D4.5 – Final COMPRISE SDK prototype and documentation
Version Nº:	1.0
Lead beneficiary:	ASCO
Author(s):	Gerrit Klasen (ASCO)
Reviewers:	Denis Jouvet (INRIA), Youssef Ridene (NETF)
Submission date:	31/05/2021
Due date:	31/05/2021
Type[1]:	DEM
Dissemination level[2]:	PU

Document history			
Date	Version	Author(s)	Comments
04/05/2021	0.1	Carolin Wübbe	First draft of ToC
24/05/2021	0.2	Gerrit Klasen	First draft of the content
25/05/2021	0.3	Carolin Wübbe	Final draft
31/05/2021	1.0	Emmanuel Vincent & Akira Campbell	Final version reviewed by the Coordinator and the project manager

[1] **R**: Report, **DEC:** Websites, patent filling, videos; **DEM:** Demonstrator, pilot, prototype; **ORDP:** Open Research Data Pilot; **ETHICS:** Ethics requirement. **OTHER:** Software Tools

[2] **PU:** Public**; CO:** Confidential, only for members of the consortium (including the Commission Services)

Document summary

This deliverable (Deliverable D4.5) is the fifth and final deliverable of Work Package 4 which is about cost-effective multilingual voice interaction. It describes the final version of the COMPRISE software development kit (SDK) prototype, which is aligned to the goals of Task T4.2. This deliverable also includes the essential documentation of the various components and provides links to more detailed and up-to-date online documentation.

The COMPRISE SDK integrates and interfaces multiple algorithms, application programming interfaces (APIs) and tools developed within COMPRISE. It facilitates the development of multilingual, voice-enabled applications by providing developers with a language abstraction for voice interaction and interfaces to easily access the desired functionalities. Furthermore, the COMPRISE SDK allows developers to compile the output across multiple platforms and to create executable Apps for both Android and iOS.

The COMPRISE SDK consists of three components: the COMPRISE Personal Server, the COMPRISE App Wizard, and the COMPRISE Client Libraries.

This document focuses on three main aspects. Firstly, it gives an explanation about the interdependencies between the three components and how they interact in the COMPRISE SDK system environment. Secondly, it explains the changes in the architecture and the technical realization that were required when compared to Deliverable D4.1 (SDK Software Architecture) and Deliverable D4.3 (Initial COMPRISE SDK prototype). Thirdly, it provides guidelines for App developers on how to install and use the COMPRISE SDK and which functionality is included into their current Apps when using the results of the COMPRISE project.

Table of contents

1 Introduction

1.1 The COMPRISE SDK

As part of Work Package 4 "Cost-effective multilingual voice interaction", Task 4.2 is dedicated to developing the COMPRISE SDK and ran from M6 to M30. Due to the high potential of unforeseen integration problems of multiple components, the development and definition of the architecture started three months earlier, already the beginning of M3. The purpose of the COMPRISE SDK is to both ease the development of privacy-aware, multilingual, voice-enabled Apps, and to reduce the duration and therefore the costs of the development.

After the initial draft of the architecture in Deliverable D4.1 (submitted to the European Commission on November 29, 2019) and a first implementation as outcome of Deliverable D4.3 (submitted to the European Commission on August 31, 2020), Deliverable D4.5 is the final Deliverable write up of the COMPRISE SDK, which consists of three components:

- The first component, the COMPRISE Personal Server, is as an extension of the second component, the COMPRISE Client Libraries. The COMPRISE Personal Server will execute computationally intense and/or data intense tasks in the voice data processing chain.
- The COMPRISE Client Libraries are installed on the user's device and connect to the COMPRISE Personal Server via an API.
- The third component, the COMPRISE App Wizard, is used by developers to connect their App with the COMPRISE components and to apply additional configurations regarding, e.g., Spoken Language Understanding models.

Compared to the previous Deliverable D4.3, multiple improvements have been realized. In summary:

- The COMPRISE Personal Server does not embed Speech-To-Text and Text-to-Speech models for all languages anymore. Instead, it includes a method that downloads only the required model(s) from the COMPRISE Cloud Platform once and for all. This reduces the necessary storage space and makes the COMPRISE Personal Server more optimised and efficient. Furthermore, Privacy-Driven Speech-Transformation features, the COMPRISE Voice Transformer and the COMPRISE Text Transformer, which change the voice identity and remove sensitive details from text, are now integrated into the COMPRISE Personal Server.
- The COMPRISE Client Libraries are now available as Web-API (NPM-library) for both Android and iOS devices. These integration tools have been regularly updated in terms of functionality, with the latest additions being streamRecording and additional models such as new language pairs for Machine Translation.
- The COMPRISE App Wizard has been extended to enrich Apps with COMPRISE functionality. The COMPRISE App Wizard now offers the developer more configuration mechanisms when selecting and setting up the models to be used in their App and also forms a central connection point for all relevant documentation of the COMPRISE SDK.

The COMPRISE SDK in its final version has multiple interdependencies with components developed in other Work Packages (see Figure 1).

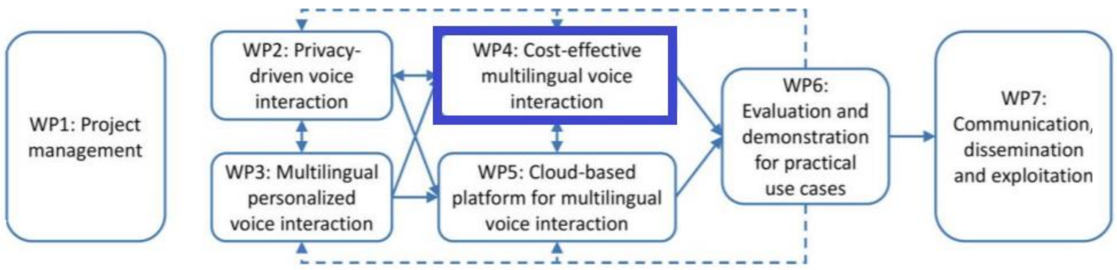

WP1: Project management
WP2: Privacy-driven voice interaction
WP3: Multilingual personalized voice interaction
WP4: Cost-effective multilingual voice interaction
WP5: Cloud-based platform for multilingual voice interaction
WP6: Evaluation and demonstration for practical use cases
WP7: Communication, dissemination and exploitation

Figure 1: COMPRISE Work Packages interdependencies.

It integrates and/or makes use of the components of Work Package 2 and Work Package 3, namely Privacy-Driven Transformation Libraries (Deliverable D2.3), Multilingual Interaction Library (Deliverable D3.3), and Personalized Learning Library (Deliverable D3.4). Those components are part of the Training Branch (blue arrows in Figure 3), while other components needed for the voice data processing chain are not in the scope of the project and taken from other solutions (orange arrows, Figure 3). Work Package 5 delivers the COMPRISE Cloud Platform, which hosts relevant models for the voice-enabled Apps at runtime, while the Apps will send privacy transformed speech and text data back for further data curation and model training. The COMPRISE SDK with the integrated functionality, together with the COMPRISE Cloud Platform, will be used by Work Package 6, where multiple demonstrators prove the functionality of the COMPRISE environment.

1.2 Content of the document

The structure of the rest of this document is the following. Section 2 explains the details of the COMPRISE SDK, its main components and how they interact with each other. Namely, technical details and explanations for the COMPRISE Personal Server, the COMPRISE App Wizard, and the COMPRISE Client Libraries will follow. Where suitable, changes compared to Deliverable D4.1 and Deliverable D4.3 are mentioned and justified. Section 3 addresses these components again, but more from the developers' and end-users' perspective, to explain which steps are needed to make the system run. As the COMPRISE SDK consists of multiple repositories, Section 4 provides an overview of the sources and the associated documentation. As the final COMPRISE SDK prototype, the current solution still offers possibilities to be improved. Section 5 gives an outlook of which improvements are likely to be carried. The deliverable ends with a conclusion in Section 6.

2 The COMPRISE SDK Prototype

As stated in Section 1, the COMPRISE SDK consists of three components, namely:

1. the COMPRISE Personal Server
2. the COMPRISE App Wizard
3. the COMPRISE Client Libraries

To attach COMPRISE functionality to individual projects (Apps), developers shall use these components in the above order, with the steps described in Section 3.

Before explaining those components in detail, a broad summary of the development and execution process with the COMPRISE SDK is given in Figure 2, including all SDK components and the COMPRISE Cloud Platform.

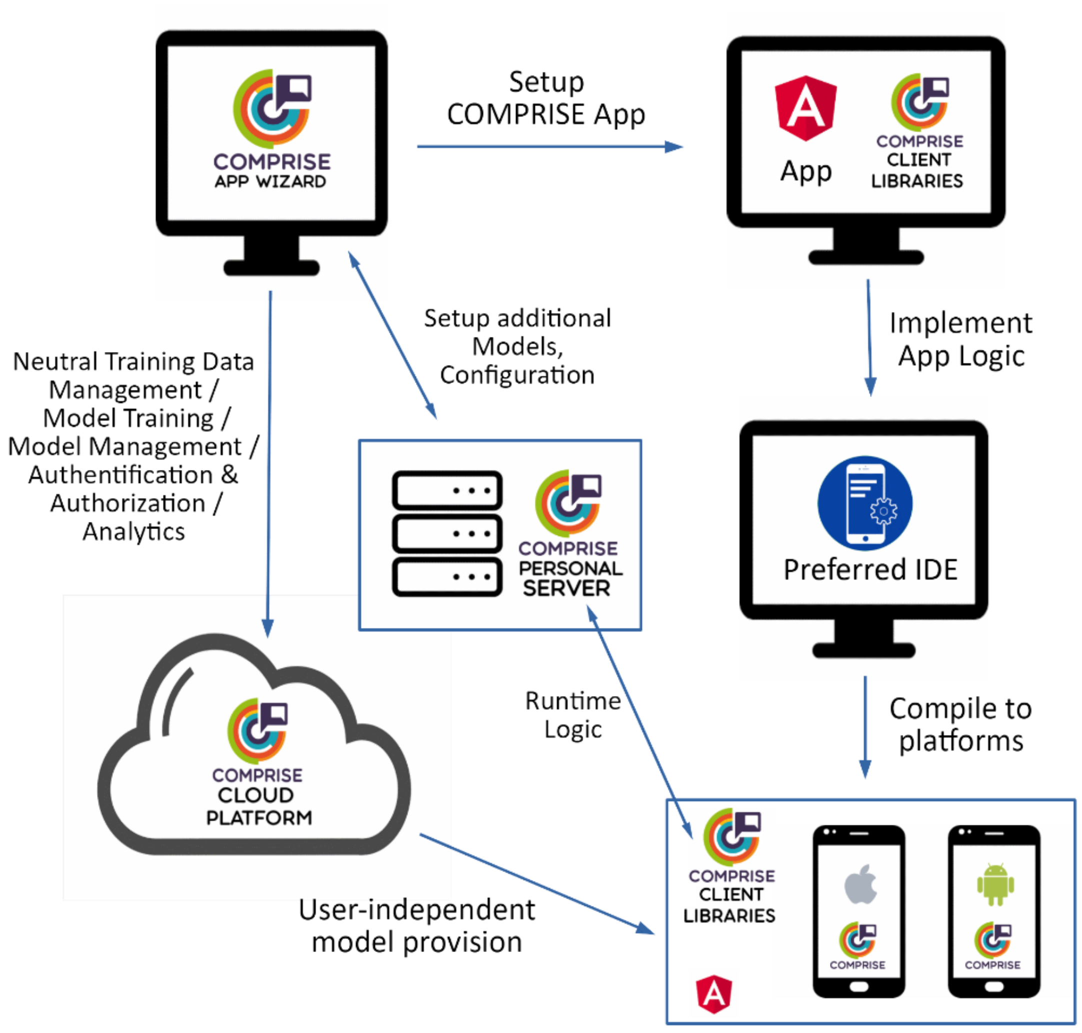

COMPRISE APP WIZARD
Setup COMPRISE App
App
COMPRISE CLIENT LIBRARIES
Setup additional Models, Configuration
Implement App Logic
Neutral Training Data Management / Model Training / Model Management / Authentification & Authorization / Analytics
COMPRISE PERSONAL SERVER
Preferred IDE
Runtime Logic
Compile to platforms
COMPRISE CLOUD PLATFORM
COMPRISE CLIENT LIBRARIES
COMPRISE
COMPRISE
User-independent model provision

Figure 2: COMPRISE SDK development process.

The overall flow is as follows:

1. The COMPRISE Personal Server is setup to execute In-App-functionalities.
2. The COMPRISE App Wizard is setup to insert the COMPRISE Personal Server address and do additional configurations.
3. The COMPRISE App Wizard creates an Application Bucket for the App within the COMPRISE Cloud Platform. In the Wizard itself, the developer also can access the COMPRISE Cloud Platform via the User Interface for Model training, etc.
4. The App Wizard extends the App with the COMPRISE Client Libraries (= APIs to Personal Server) and the COMPRISE Personal Server address.
5. The developer uses the COMPRISE Client Libraries installed to efficiently use COMPRISE functionality within the App.
6. The developer deploys the App which is now voice-enabled, multilingual and privacy aware.
7. The App at runtime is served with a user-independent model from the COMPRISE Cloud Platform, and sends privacy-transformed speech and text data back for training.
8. The COMPRISE Personal Server is connected to the App to execute computationally intense processes.

Using the current version of the COMPRISE SDK, the end-users of the Apps created by the developer do not need to configure any of the SDK components. All configuration (which Spoken Language Understanding models are needed within the App, what is the URL of the COMPRISE Personal Server, etc.) is done by the developer. This addresses the situation when the company or institution exploiting the App (e.g., a hospital) runs a single instance of the COMPRISE Personal Server on a trusted computer located within their premises that serves all end-users. In that situation, the end-users will not suffer privacy issues, as long as the COMPRISE Voice Transformer and COMPRISE Text Transformer are used and the company or institution exploiting the App complies with the General Data Protection Regulation[3] (GDPR), takes proper measures to ensure a high level of security for the COMPRISE Personal Server, and specifies an intended usage of the data that is not too broad.

In the situations when these conditions are not met, the ultimate goal would be for the functionality of the COMPRISE Personal Server to be within the end-user's device, or for the trusted-computer running the COMPRISE Personal Server to be located within each end-user's household. However, with the current prototype, the end-users would need a level of technical experience and knowledge that cannot be expected from the average end-user. Enabling the end-users to install and run their own instance of the COMPRISE Personal Server and connect it to the Apps in a quick and easy way is part of future work.

2.1 The COMPRISE Personal Server

The COMPRISE Personal Server is an extension to the COMPRISE SDK environment, identified and implemented in the context of Deliverable D4.3, where the first version of the COMPRISE SDK has been released.

As can be seen in the white area of Figure 3, multiple small- to medium-sized components, which are part of the COMPRISE Client Library (see Section 2.3), need to run on the user's device.

In the course of the project, we have come to the conclusion that the technologies chosen for the voice interaction chain are not yet optimal, due to hardware and software issues, for use on a smartphone environment. This was problematic since it prevented some of the COMPRISE Client Library components from functioning. The provision of a global server operated by a voice technology company and taking over these services for all Apps would have been a solution regarding the storage and computational power needed. Still, this would force end-users to trust the voice technology company about the utilization of their sensitive data while processing their voice input. As COMPRISE aimed to avoid exactly this, relying on such a global server is not an option.

[3] https://gdpr.eu/

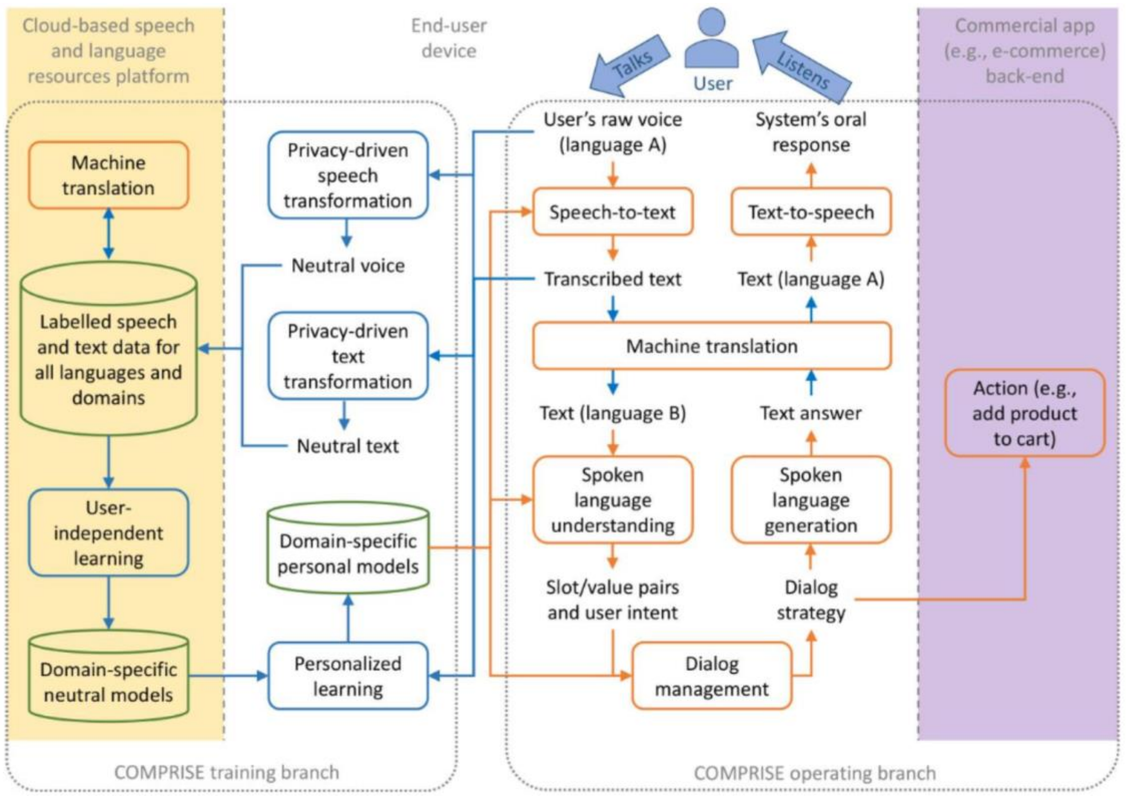

Cloud-based speech and language resources platform
End-user device
User
Talks
Listens
Commercial app (e.g., e-commerce) back-end
Machine translation
Labelled speech and text data for all languages and domains
User-independent learning
Domain-specific neutral models
Privacy-driven speech transformation
Neutral voice
Privacy-driven text transformation
Neutral text
Domain-specific personal models
Personalized learning
User's raw voice (language A)
System's oral response
Speech-to-text
Text-to-speech
Transcribed text
Text (language A)
Machine translation
Text (language B)
Text answer
Spoken language understanding
Spoken language generation
Slot/value pairs and user intent
Dialog strategy
Dialog management
Action (e.g., add product to cart)
COMPRISE training branch
COMPRISE operating branch

Figure 3: Data flow of voice interaction system within COMPRISE.

As a result, besides the initial COMPRISE SDK Components "App Wizard" and "Client Libraries", the "Personal Server" has been instantiated.

The COMPRISE Personal Server itself is a server packaged within a portable container. By using a server instead of the end-user's device, it became possible to solve previous issues such as limited computational power and/or storage on smartphones, and code incompatibilities between Android/iOS and voice processing chain implementations. The COMPRISE Personal Server can be run by the company or institution exploiting each App, so that only this company or institution accesses the users' sensitive data for this App for a narrow-intended usage (as opposed to a voice technology company accessing data across multiple Apps for a broader usage). Alternatively, it could also be run by each end-user, so that the end-user is the only person having access to his/her data. In comparison to normal servers, maintainers of the system do not need a backend server to run the COMPRISE Personal Server. It is containerized with the help of Docker[4] and can also run on local desktop computers using Windows or macOS, which gives the user more flexibility.

The COMPRISE Client Libraries are still needed (see Section 2.3), and they are now functioning as API connections between the COMPRISE Personal Server logic and the smartphone App. Each component involved is able to contact the server to execute the required behaviour, such as Privacy-Driven Voice and Text Transformation, and is receiving the result back in a RESTful way (Figure 4).

In the current version of the COMPRISE SDK, all the components are running on the COMPRISE Personal Server, leaving the COMPRISE Client Libraries responsible as an API. This ensures the compatibility with both Android and iOS. As mentioned later in

[4] https://www.docker.com/

Section 5, in the longer term, these components are to be developed to run on the user's device as initially planned.

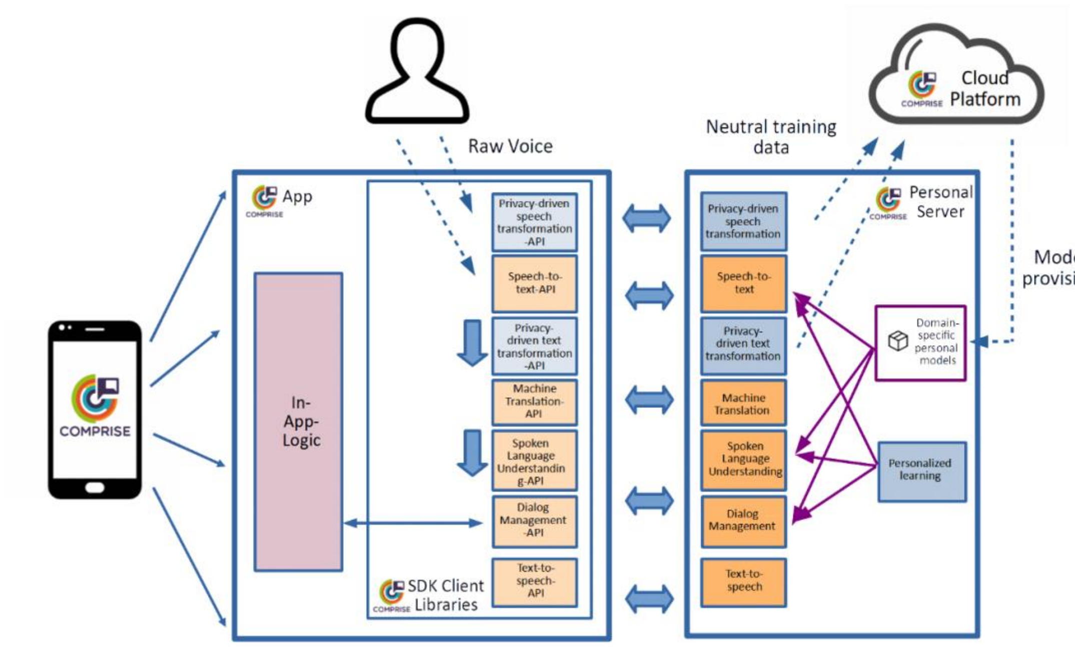


Figure 4: Interaction between the COMPRISE Personal Server and COMPRISE Client Libraries.

The addition of the COMPRISE Personal Server raised a new concern: opening access to a specific URL/IP address so that the COMPRISE Personal Server can communicate with the end-user's smartphone. This newly required information is attached to the App and the COMPRISE Client Libraries from within the COMPRISE App Wizard (see Section 2.2).

The provision of this URL is not a problem regarding backend servers, as they usually provide fixed IP addresses. However, in the development phase, developers can also setup the COMPRISE Personal Server on a local desktop machine, which usually cannot be accessed from outside their local network. These local machines can normally only be accessed from within their "localhost" environment. To prevent this, partner Ascora has set up a proxy server using the pagekite[5] framework. Pagekite allows securely exposing a local server environment to the wider internet with the help of HTTPS tunnels.

As shown in Figure 5, when a developer initializes a Personal Server instance, a URL will be generated in the form "xxxxx.comprise-pagekite2.ascora.eu". The "xxxxx" is a placeholder for a randomly generated character sequence. This enables the generation of an adequate quantity of servers, i.e., one per developer interested in the COMPRISE solution. Once the developer publishes their App, the functions triggered by COMPRISE Client Libraries are processed through a secure HTTPS tunnel to Ascora's Proxy Server. The request is never saved on the proxy server and is directly forwarded to the desired Personal Server instance.

[5] https://pagekite.net/

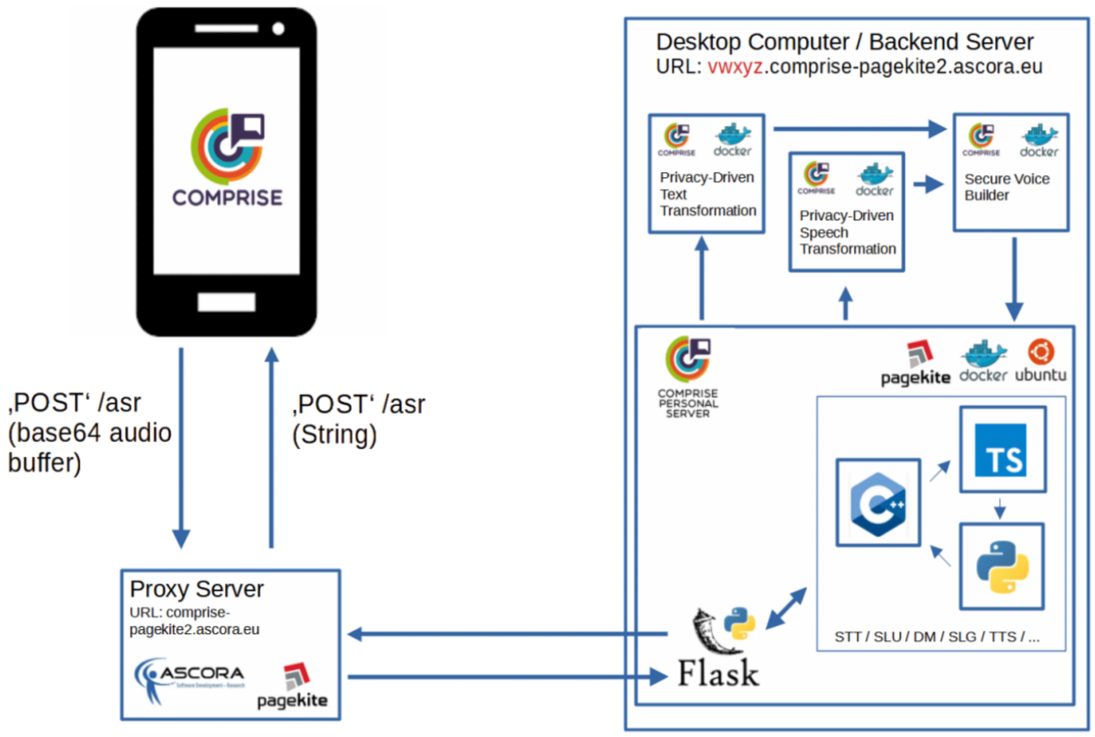

COMPRISE
‚POST‘ /asr (base64 audio buffer)
‚POST‘ /asr (String)
Proxy Server
URL: comprise-pagekite2.ascora.eu
ASCORA
pagekite
Desktop Computer / Backend Server
URL: vwxyz.comprise-pagekite2.ascora.eu
Privacy-Driven Text Transformation
Privacy-Driven Speech Transformation
Secure Voice Builder
COMPRISE PERSONAL SERVER
pagekite
docker
ubuntu
TS
STT / SLU / DM / SLG / TTS / ...
Flask

Figure 5: Internal COMPRISE Personal Server Logic and interaction.

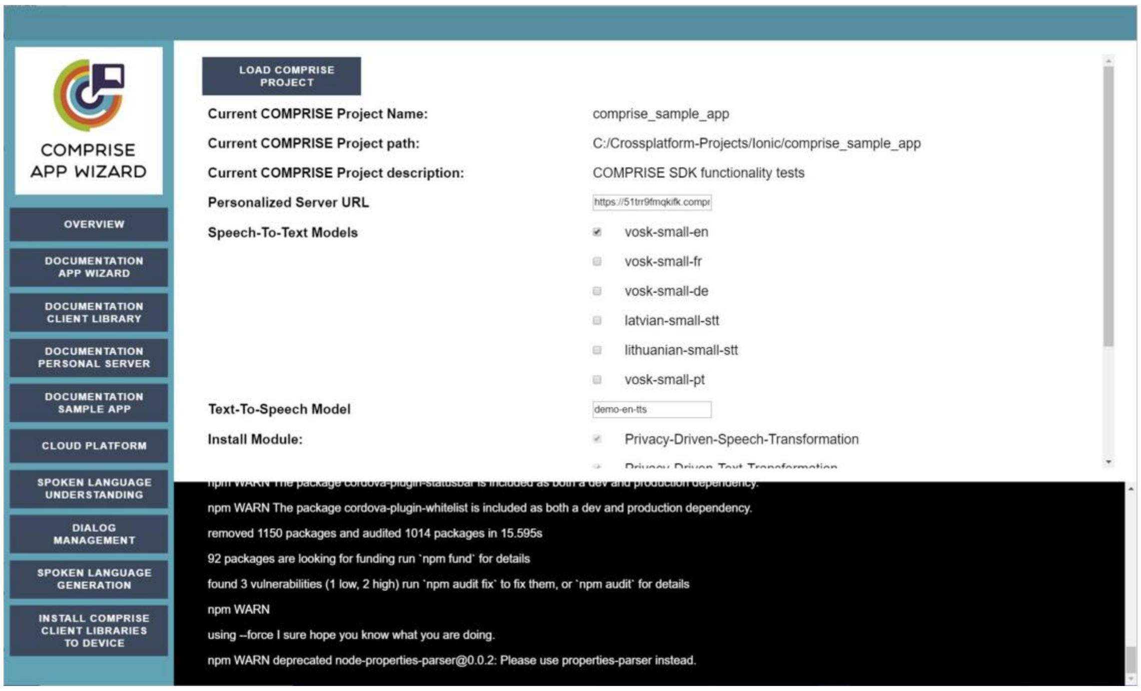
Compared to D4.3, the following improvements have been brought:

- The first prototype of the COMPRISE SDK described in Deliverable D4.3 allowed the device to communicate with the COMPRISE Personal Server using the ngrok[6] framework within the Docker container itself. Ngrok opened the local network for the container, which was directly connected to the App. In the course of developing the final version of the COMPRISE SDK, ngrok was replaced with mysocket.io[7] due to its better scalability.
- To enable a stable integration, the consortium had close contact with the author of mysocket.io, who unfortunately was not able to guarantee the maintenance of the this relatively new framework in the near future, let alone the mid- to long-term future. This required the replacement of ngrok by Pagekite and general changes in the architecture which resulted in the Proxy Server settings described above. The use of Pagekite enables a large number of Personal Servers in parallel with a proxy framework that can be supported in the long term.
- Detailed documentation can be located in the corresponding repository (see Section 4).

2.2 The COMPRISE App Wizard

The COMPRISE App Wizard is a helper tool for software developers, which allows them to equip their smartphone solutions with COMPRISE functionality and therefore include the project’s benefits, i.e., voice-enabled functionalities in a multilingual and privacy-aware way.

[6] https://ngrok.com/
[7] https://www.mysocket.io/


COMPRISE APP WIZARD
OVERVIEW
DOCUMENTATION APP WIZARD
DOCUMENTATION CLIENT LIBRARY
DOCUMENTATION PERSONAL SERVER
DOCUMENTATION SAMPLE APP
CLOUD PLATFORM
SPOKEN LANGUAGE UNDERSTANDING
DIALOG MANAGEMENT
SPOKEN LANGUAGE GENERATION
INSTALL COMPRISE CLIENT LIBRARIES TO DEVICE
LOAD COMPRISE PROJECT

Current COMPRISE Project Name:	comprise_sample_app
Current COMPRISE Project path:	C:/Crossplatform-Projects/Ionic/comprise_sample_app
Current COMPRISE Project description:	COMPRISE SDK functionality tests
Personalized Server URL	https://51trr9fmqkifk.compr
Speech-To-Text Models	✓ vosk-small-en
	☐ vosk-small-fr
	☐ vosk-small-de
	☐ latvian-small-stt
	☐ lithuanian-small-stt
	☐ vosk-small-pt
Text-To-Speech Model	demo-en-tts
Install Module:	✓ Privacy-Driven-Speech-Transformation

```
npm WARN The package cordova-plugin-whitelist is included as both a dev and production dependency.
removed 1150 packages and audited 1014 packages in 15.595s
92 packages are looking for funding run `npm fund` for details
found 3 vulnerabilities (1 low, 2 high) run `npm audit fix` to fix them, or `npm audit` for details
npm WARN
using --force I sure hope you know what you are doing.
npm WARN deprecated node-properties-parser@0.0.2: Please use properties-parser instead.
```

Figure 6: COMPRISE Application Wizard.

In the scope of the project, the COMPRISE App Wizard is suitable for all smartphone applications developed with the Ionic Framework and the Angular Framework. This combination of technologies has been chosen to make COMPRISE compatible with cross platform technology. The Apps enriched by COMPRISE therefore are and will still be deployable on multiple operating systems (i.e., iOS and Android), ensuring that developers are not required to develop the code twice in the native language of each operating system, but stick to one code base instead. This will shorten the development time, and as a consequence reduce the related costs, which is in line with the targets of Work Package 4.

Multiple items both can and need to be specified within that component. As further described in Section 3, developers will use the COMPRISE App Wizard after having set up the COMPRISE Personal Server. At first the App of the developer is not connected to any COMPRISE related component. In a first setup, developers are able to select their App and load it into the COMPRISE App Wizard. This will inform the software which App to extend with the COMPRISE components.

After the App has been selected, a storage bucket will be created within the COMPRISE Cloud Platform, which is dedicated to the App. It will be used to receive the anonymized speech and text data of the users at runtime. Once enough data is collected and a domain-specific, user-independent model has been trained on this data, the App will be able to download the model from this bucket. More information about the COMPRISE Cloud Platform is available in Deliverable D5.4 “Initial platform demonstrator” (submitted to the European Commission on November 30, 2020) and Deliverable D5.5 “Final platform demonstrator and updated data protection and GDPR requirements” (submitted to the European Commission on May 31, 2021). The COMPRISE Cloud Platform itself is also accessible by developers/annotators at any time via the COMPRISE App Wizard.

Once the COMPRISE Personal Server and the COMPRISE App Wizard are setup, the connection URL generated by the COMPRISE Personal Server can be inserted into the

COMPRISE App Wizard. When the developer completes all the COMPRISE App Wizard configurations and the compilation starts, the component will attach the URL to the App. Within the App, the COMPRISE Client Libraries can then start referring to it. The API components can properly address the COMPRISE Personal Server to execute the needed functionalities. This is the point where the backend services/trusted local computer and the smartphone App get their actual connection.

In addition, the COMPRISE App Wizard will be needed to choose or configure models from the Operating Branch, namely for Speech-To-Text (STT), Spoken Language Understanding (SLU) and Text-To-Speech (TTS).

Regarding STT and TTS, potentially, the COMPRISE Personal Server could host several models to choose from. This would result in the corresponding Docker container becoming extremely large and taking a long time to download, in addition to the large amount of free storage space needed. As a solution, the STT models have been moved from the COMPRISE Personal Server to the COMPRISE Cloud Platform. This results in a smaller server container. Once the developer triggers the installation within the COMPRISE App Wizard, the COMPRISE Cloud Platform sends the chosen STT/TTS models to the COMPRISE Personal Server, which will trigger the download of the models needed from the COMPRISE Cloud Platform. They will then be available for in-app-usage for the end-users. For TTS, a similar behaviour is in progress and will be completed before the end of the project.

To setup the App use case scenario for the Natural Language Processing Chain (NLP), the COMPRISE App Wizard provides access to partner TILDE's Dialog System training "TILDE.AI". It provides all sources needed to properly train Intent detection,[8] to define a real use case scenario[9] and to deliver potential answers based on the identified intent.[10] All documentation needed is available within the Tilde.AI training PDF[11] and/or video[12].

Besides all configurations, the COMPRISE App Wizard is the central connection point where developers can find all relevant documentation of the related components. It is linked for the COMPRISE SDK Component repositories as well as for the COMPRISE Sample App.

After the configuration, the developer can start the build process, where COMPRISE will be attached to the App itself. The COMPRISE Personal Server connection, as mentioned earlier, will be made, as the URL will be noted as part of the App. The COMPRISE Client Libraries, which contain multiple APIs, will be installed one by one and make use of the provided URL. To ensure proper functionality, the COMPRISE App Wizard will remove old native Android/iOS code from the targeted Ionic App and will generate a fresh new version with COMPRISE results integrated.

Compared to Deliverable D4.3, the following functionalities have been added:

- The initial version, which offered a button referencing the COMPRISE App Wizard documentation, has been extended by multiple other references. The COMPRISE App Wizard is supposed to be a connection point for most of the components of the project's ecosystem. As a result, multiple buttons have been added, linking

[8] https://botdashboard.tilde.ai/Intents
[9] https://botdashboard.tilde.ai/Model
[10] https://botdashboard.tilde.ai/Lang
[11] https://gitlab.inria.fr/comprise/comprise_app_wizard/-/blob/master/doc/Tilde.AI%20Training.pdf
[12] https://www.youtube.com/watch?v=fVciLmr1VDI&feature=youtu.be

documentations for the COMPRISE App Wizard itself, but also for the COMPRISE Client Libraries, the COMPRISE Personal Server, as well as for the COMPRISE Sample App project, which serves as a starting point for new potential users.

- Models for Speech-To-Text are now customizable and selectable, allowing the developer to define the languages supported within the App. All the choices are downloaded within the COMPRISE Personal Server once the developer starts to install the COMPRISE Client Libraries to the App. The location has been moved from the COMPRISE Personal Server to the COMPRISE Cloud Platform.
- Similar behaviour is currently implemented for Text-To-Speech models. A default selection of TTS models are currently present for all languages supported within the COMPRISE Personal Server. Before the project ends, they will be moved from the COMPRISE Personal Server to the COMPRISE Cloud Platform as well. From there, they will be downloaded when needed to keep the COMPRISE Personal Server Docker container size small and to carry only the components needed.
- As Deliverable D4.3 focused on setting up an example application with COMPRISE functionality, the final prototype focused on integrating COMPRISE to bigger, individual Apps.
- Detailed documentation can be located in the corresponding repositories (see Section 4).

2.3 The COMPRISE Client Libraries

As mentioned in the previous section, the initial architecture of the COMPRISE Client Libraries as stated in Deliverable D4.1 had been foreseen to integrate multiple components (see Figure 2) of the Operating and Training Branch directly on the user's device. While preparing the first version of Deliverable D4.3, which contains the first realization of the COMPRISE SDK prototype, it turned out that the integration of those components is not possible due to one or multiple reasons listed below:

- The code base of the component to be integrated is not executable on Android/iOS devices.
- The storage on the smartphones on average is too low to contain all of the needed functionalities.
- The computational power of smartphones is too low to deliver an efficient performance.
- The integration mechanisms exist but are too complex to realize within the scope of the work package and/or project.

For the final version of the COMPRISE Client Libraries, the following components have been inherited. The explanations given below both explain the possible calls done by the COMPRISE Client Libraries API itself and the functionality executed within the COMPRISE Personal Server.

2.3.1 Operating Branch

Speech-To-Text
The Speech-To-Text (STT) component is the first sub-component in the process chain. It receives the voice input, normally recorded by a user within the App.

In the STT library within the COMPRISE Client Libraries, an NPM library called RecordRTC[13] is responsible for recording the audio stream, which will be transmitted to the COMPRISE Personal Server. There, an offline-compatible Framework called Vosk API,[14] which is based on Kaldi,[15] is interpreting the data stream and translates it into a textual representation.

The models being used by Vosk API depend on the developer's configuration within the COMPRISE App Wizard. Depending on the supported languages chosen, the COMPRISE Personal Server will be informed to download a suitable STT model from the COMPRISE Cloud Platform, when in parallel the COMPRISE Client Libraries are installed to the App. This behaviour is intended to keep the data size of the COMPRISE Personal Server as small as possible.

The STT library, at the time of submitting Deliverable D4.5, allows you to stop and start audio recording by a button press, or by start and stop speaking. The choice between these methods is determined by the developer of the App.

Additionally, a Cordova[16] plugin for STT[17] is currently in progress, with the target to bring back as much automatic speech recognition functionality as possible back onto the user's device, as initially desired.

Machine Translation

The Machine Translation (MT) API will take the textual representation of the recording and translate it into a pivot language. In COMPRISE, by default, this is English, but in general this can be any language. The intention behind this is that the Spoken Language Processing components (Spoken Language Understanding, Dialogue Management, Spoken Language Generation) behind only need to be trained on the pivot language, but still allow any input language, as long as it is translated.

The MT API will transmit the text to partner TILDE`s Machine Translation Tool "LetsMT", [18] where the native language will be translated into the desired pivot language, which can be further processed by following NLP components.

This step will be required twice within the voice data processing chain. A second translation will take place after the intent of the user input has been interpreted, processed internally within the App, and a suitable answer has been generated. The answer, still in pivot language, needs to be translated back into the native representation, so that the Text-To-Speech entity is able to generate an understandable audio output.

The MT library, at the time of submitting Deliverable D4.5, allows the translation method for the language pairs provided by TILDE.

[13] https://github.com/muaz-khan/RecordRTC
[14] https://github.com/alphacep/vosk-api
[15] https://kaldi-asr.org/
[16] Cordova is a framework connecting native OS code to Angular apps: https://cordova.apache.org/.
[17] https://www.npmjs.com/package/cordova-plugin-comprise-text-to-speech
[18] https://www.letsmt.eu/

Spoken Language Understanding
The Spoken Language Understanding (SLU) API sends the textual representation (in pivot language) to partner TILDE's Dialog System training "TILDE.AI".[19] As part of the COMPRISE App Wizard configurations, the developer has defined multiple potential word combinations which each are connected to a specific ID, the intent. The voice input "Good morning" possibly would be interpreted as ID "greeting", which then will be delivered back to the application. The SLU library, at the time of submitting Deliverable D4.5, provides the intent-detection method.

Dialogue Management / Spoken Language Generation
Dialogue Management (DM) and Spoken Language Generation (SLG) often are handled separately, but within the scope of COMPRISE, they form one library/API. Regarding Dialogue Management (how to proceed with a certain intent detected), there is no library functionality to be generalized, as the application developer decides on how to proceed within the internal code. The received intent provided by the Spoken Language Understanding Library can be handled directly within the App's logic, which is not COMPRISE related any more, like automatically generating an order, when receiving the intent "order". Within the NLP-related libraries themselves, all conversations with TILDE.AI are taking place using the Microsoft Bot Framework.[20] For Spoken Language Generation, the library transmits the text to TILDE.AI, where the developer has also defined how to answer each intent. The callback will be one sentence (in pivot language) out of potentially multiple possibilities for one intent. The SLG library, at the time of submitting Deliverable D4.5, provides multiple methods. It is possible to open a new, fresh conversation and to retrieve an existing conversation (important for chained intents). It enables you to send messages with an intent in it and to retrieve a suitable answer within a given conversation.

Text-To-Speech
The Text-To-Speech (TTS) API takes the textual representation of the answer generated by SLG in the native language, after it has been translated there from pivot language by the MT module. It transforms it back to an audio stream, which will be played within the library by a simple HTML5 audio element. Depending on the language chosen, different methods are currently being called. Within COMPRISE, the following 6 languages are supported: English, French, German, Latvian, Lithuanian, and Portuguese.

[19] https://botdashboard.tilde.ai/
[20] https://directline.botframework.com

English, French and German are handled by Mozilla TTS.[21] The English models provided by Mozilla TTS are used without modification, while the French and German models have been retrained for better quality. They are stored within the COMPRISE Personal Server and transform text input into an audio stream, which is transmitted back to the application.

Latvian, Lithuanian and Portuguese are currently handled by native Google TTS, due to the lack of a suitable, publicly available data set for training the models. As this is not privacy-preserving the way COMPRISE aims to handle it, this solution is temporary and meant to be replaced by Mozilla TTS once such data sets become available.

The TTS library, at the time of submitting Deliverable D4.5, provides the TTS transformation method.

Additionally, a Cordova plugin for TTS[22] is currently in progress, with the target to bring back as much functionality as possible into the user's device, as initially desired.

2.3.2 *Training Branch*

Privacy-Driven Speech Transformation

The Privacy-Driven Speech Transformation (PDST) API processes the user's raw voice input in the COMPRISE Personal Server. There, the voice will be transformed by another, locally running Docker container, which is responsible for the Privacy-Driven Speech Transformation. The COMPRISE Personal Server actually consists of three Docker linked containers: the Personal Server itself, handling most of the project's backend code within that container, and two additional containers for the Privacy-Driven Speech and Text Transformations. Further description can be found in Deliverable D2.3,[23] which delivers the final version of Speech Transformation.

It will transform the audio in two steps.

1) The speech signal will be transformed into an anonymized signal with different voice characteristics, ensuring that nobody can identify the speaker's identity any longer.
2) Sensitive information within the sequence, like credit card numbers, names, or current locations will be completely cut out.

The output will be sent to the COMPRISE Cloud Platform. The component therefore will ensure that within the COMPRISE Cloud Platform, model training is still possible as the original context of the message remains, but private information is completely removed or modified.

The PDST library, at the time of submitting Deliverable D4.5, provides the speech transformation method.

[21] https://github.com/mozilla/TTS
[22] https://www.npmjs.com/package/cordova-plugin-comprise-speech-to-text
[23] https://www.compriseh2020.eu/files/2021/02/D2.3.pdf

Privacy-Driven Text Transformation
The Privacy-Driven Text Transformation (PDTT) API is also a research result reported in Deliverable D2.3, which will receive the raw user input in a textual representation (after STT). It will transform the text in a way that any sensitive information is not removed but replaced by a similar entity of a similar context. For example, a user stating that he/she currently is in Berlin will result in a sentence stating that he/she is in London or another city. The output will be sent to the COMPRISE Cloud Platform. The component therefore will ensure that within the COMPRISE Cloud Platform, model training is still possible as the original context of the message remains, but private information is completely removed or modified. The PDTT library, at the time of submitting Deliverable D4.5, provides the text transformation method.

COMPRISE Cloud Platform
The Cloud Platform (CP) API transmits the privacy-transformed neutral text and speech data to the COMPRISE Cloud Platform. There, the corresponding (manual or automatic) labels are stored. The COMPRISE Cloud Platform allows developers and annotators to upload, store and manage data and labels and train or access large-scale user independent models trained on the collected data. Secondly, related data is generated and uploaded at run-time, when the application user uses the App. The COMPRISE Cloud Platform functionality includes secure cloud-based data and model storage, scalable and dynamic cloud-based high-performance computing, APIs for continuous data upload and occasional model download, and general features (user interface, authentication, usage analytics, etc.) and procedures for data labelling and curation. The COMPRISE Cloud Platform itself is accessible, among other ways, via the COMPRISE App Wizard. The CP library, at the time of submitting Deliverable D4.5, provides transmission methods for (previously anonymized) text and speech entities. Note that the models needed at run-time (like STT and SLU) are not part of this library, but part of the behaviour between the COMPRISE App Wizard and the COMPRISE Personal Server (as mentioned in Section 2.2). When the developer chooses the relevant models in the COMPRISE App Wizard, they are downloaded on the COMPRISE Personal Server. When the user starts the App, the smartphone App connects to the server, containing the relevant models.

Personalized Learning
The Personalized Learning (PL) component is not part of the final version of the COMPRISE SDK, in contrast to the initial planning reported in Deliverable D4.1. The

way Personalized Learning can be realized is not depending on a "Personalized Learning" component itself. The user-independent models received by the COMPRISE Personal Server would be handled as part of four different concepts:

1) STT Acoustic Model Personalisation is part of the STT component itself. The STT component based on Vosk API performs speaker adaptation by means of an i-vector computed on-the-fly. In addition, a solution was developed for accent robustness in Work Package 3 and made available as standalone code in the COMPRISE GitLab. This solution was successfully validated on accented English, but gave mixed results on accented Latvian data. Accented data for French, German, Lithuanian and Portuguese were found to be inexistent or too scarce to develop and validate the solution in these languages. Therefore, the consortium decided not to integrate a solution that was not successfully approved for other languages in the final COMPRISE SDK.

2-4) As presented in Deliverable D3.4, STT Language Model Personalisation, SLU Model Personalisation and DM Model Personalisation require a lot of personal data for only a small improvement, resulting in an unneeded integration.

All of the components are realized as installable APIs on NPM[24] and are publicly available in GitLab[25] for further contributions and discussions with interested developers and users. The URLs to both GitLab and NPM of each component can be found in the documentation overview in Section 4.

As the Apps to be potentially enriched within the scope of the project are based on the Angular framework, the libraries mentioned above are also published within NPM repositories. NPM hosts libraries which can be installed on Angular Apps.

Once the developer has setup NodeJS[26] on their OS, the latest version of the desired library can be installed via the following command (here with the example of the Privacy-Driven Text Transformation):

```
npm i –save comprise_privacy_driven_text_transformation@latest
```

In normal circumstances, a developer is not required to perform this step. The COMPRISE App Wizard takes over this part when extending the App with COMPRISE functionality.

As for the GitLab resources, Section 4 also gives references for the NPM registries.

Compared to Deliverable D4.3, the following improvements have been brought:

- Regarding the Training Branch components, the COMPRISE SDK Client Libraries are now connected to the latest versions of Privacy-Driven Speech Transformation (including voice anonymization and removal of sensitive content), Privacy-Driven Text Transformation and the COMPRISE Cloud Platform.
- Updates of various libraries regarding code quality, inclusion of additional models (e.g., additional languages supported for Machine Translation).

[24] https://www.npmjs.com/
[25] https://gitlab.inria.fr/comprise
[26] https://nodejs.org/en/

- All components needed are available as Web-API/NPM-library to provide a Minimal Valuable Product (MVP) which can be integrated on both Android and iOS devices via the Angular framework. First components (STT, TTS) also are available as a Cordova[27] plugin to bring back "on-device-integration" without the need for API calls. They are aimed to be continued with other components as well, depending on compatibility.
- Detailed documentation can be located in the corresponding repositories (see Section 4).

3 How to get started

In general, software developers will mostly be interested in COMPRISE when they own one or multiple smartphone Apps they want to extend with the functionality of the COMPRISE project. The instructions below assume that the App to be extended is already present on the corresponding computer.

The three following steps are required:

1. Setup a COMPRISE Personal Server to execute the needed functionality exclusively for their Apps.
2. Execute the COMPRISE App Wizard to connect the developer's App with the freshly generated COMPRISE Personal Server resource, install COMPRISE Client Libraries to the App, additionally do customized configurations such as setting up the application scenario models, language and acoustic related models, etc.
3. Use the COMPRISE Client Libraries within an IDE of the developer's choice to add all desired behaviour to the application logic.

A developer can setup the COMPRISE SDK from a desktop computer (e.g., Windows, macOS, Linux, etc.):

- To execute the COMPRISE App Wizard and to work with it.
- To execute the COMPRISE Personal Server for debugging purposes.
- To work within the IDE of the developer's desire to integrate COMPRISE Client Libraries into the App's source code.

When a developer wants to keep the services of the COMPRISE Personal Server running within a production environment, it is recommended to use a backend server.

The documentation is available online. According to feedback from COMPRISE users, we will thrive to adapt, fix errors, and update the documentation when needed. This will continue during the whole project lifetime.

The configuration of COMPRISE environment starts with the COMPRISE Personal Server, the documentation to follow can be found here:

https://gitlab.inria.fr/comprise/comprise-personal-server

[27] https://cordova.apache.org/

COMPRISE Personal Server

Step 1: Setup Docker

The Personal Server is packaged within a container. This will enable users to download their own copy to their personal environment and work with it. This requires Docker to be setup to execute those containers. The way to install Docker differs between operating systems the user chooses to work with.

Step 2: Setup COMPRISE Personal Server

After successfully setting up Docker, the online documentation provides instructions on how the COMPRISE Personal Server runs on the developer's desktop computer or backend server.

First, a developer will need to download the containers with the help of Docker to the local filesystem. The downloaded content consists of three docker containers, but also related results being developed with the project. The download contains:

- The COMPRISE Personal Server itself
- Privacy-Driven Speech Transformation
- Secure Voice Builder
- Privacy-Driven Text Transformation

The other components of the voice data processing chain not mentioned above are integrated within the COMPRISE Personal Server. The reason behind this is to allow developers the possibility to use the Privacy-Driven Transformation Containers on their own, independently and not deeply nested within the COMPRISE Personal Server. This separation allows that.

Once completed, the documentation provides commands to:

- Update the Docker containers manually
- Run/Execute the containers
- Stop/Abort running containers

Step 3: Connect COMPRISE Personal Server with your App!

During the creation and execution of the running containers, they create a folder on the users file system, where an URL is provided. This URL is taken as a configuration parameter within the COMPRISE App Wizard to connect the developer's product with the COMPRISE Personal Server later on.

Now, the developer can switch to the COMPRISE App Wizard, the documentation to follow can be found here:

https://gitlab.inria.fr/comprise/comprise_app_wizard

COMPRISE App Wizard

Step 1: Setup App Wizard

The developer needs to setup a few dependencies to make the COMPRISE App Wizard run, such as Git[28] or NodeJS.[29]

Once completed, scripts are provided on how to install the helper tool component itself and how to start it.

Step 2: COMPRISE Project Initialization

As a first step, the project to be extended by COMPRISE is selected. Currently, this can be any Ionic[30] project developed with the Angular[31] Framework.

Step 3: Personal Server Configuration

The developer is now required to insert the URL received from the COMPRISE Personal Server earlier, so that it will be connected with the App chosen in Step 2.

Optional: Speech-To-Text / Text-To-Speech Model Configuration

The COMPRISE Personal Server, during its development, can be enriched by various STT and/or TTS models, which App developers can use for their own business. They will have the possibility to choose which ones are wanted, and which languages are supported. Once the COMPRISE Personal Server URL has been specified, the default values indicate common English language STT/TTS models.

Optional: Exclude Operating Branch Components

COMPRISE applications must include various Training Branch components to stay privacy-aware, namely the Privacy-Driven Speech/Text Transformation and the COMPRISE Cloud Platform API (see Figure 3). The Operating Branch components like STT are also available, but theoretically could be replaced by individual solutions preferred by the developer, as there are other similar solutions available in the market. They can be unchecked to prevent their installation, resulting in smaller applications without unneeded code. As a result, the developer would need to take care of these components themselves.

Step 4: Spoken Language Processing Configuration

[28] https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
[29] https://nodejs.org/en/download/
[30] https://ionicframework.com/
[31] https://angular.io/

The developer has access to an NLP platform to configure all parameters needed to define the application scenario. It contains training documentation and videos for Spoken Language Understanding, Dialogue Management and Spoken Language Generation.

Step 5: Installation of COMPRISE Client Libraries

At this point, all necessary configurations are completed and can be attached to the App. The COMPRISE Client Libraries, containing various COMPRISE functionality, will be installed to the App, as well as the COMPRISE Personal Server reference. The current progress and eventual errors can be followed within a console output within the COMPRISE App Wizard. When successful and connected to a smartphone, the App will start automatically.

As all the dependencies are now installed, developers can continue coding within the IDE of their choice and make use of all the methods provided in the API descriptions of COMPRISE Client Libraries. The API documentation can be found here:

https://gitlab.inria.fr/comprise/comprise_client_libraries

COMPRISE Client Libraries

Documentation about both Operating and Training Branch components are included:

Operating Branch APIs:

- Speech-To-Text
- Machine Translation
- Spoken Language Understanding
- Dialogue Management / Spoken Language Generation
- Text-To-Speech

Training Branch APIs:

- Privacy-Driven Speech Transformation
- Privacy-Driven Text Transformation
- COMPRISE Cloud Platform

4 Documentation Overview

The main components of the COMPRISE SDK are the COMPRISE Personal Server, the COMPRISE Client Libraries and the COMPRISE App Wizard. These components are open source and available to be used, cloned, modified and extended by interested developers. Developers can also raise issues and suggestions in the corresponding repository, where the consortium is in continuous exchange to improve the tools.

Furthermore, to ease the integration and also to provide a first-hand impression of how everything works between the COMPRISE SDK and a developer generated App, a sample App has been published.

COMPRISE SDK Components		
Component name	**Short description**	**Link to Git**
COMPRISE Personal Server	Containerized Server environment to be set up by the developer, which is able to execute multiple COMPRISE related services within the voice data processing chain.	https://gitlab.inria.fr/comprise/comprise-personal-server
COMPRISE Client Libraries	Consists of multiple API libraries to connect to the COMPRISE Personal Server. Runs on the user's device and processes the incoming results of API calls.	https://gitlab.inria.fr/comprise/comprise_client_library
COMPRISE App Wizard	Helper tool to connect COMPRISE Client Libraries of the App with the corresponding COMPRISE Personal Server, setup voice and application scenario related models and more.	https://gitlab.inria.fr/comprise/comprise_app_wizard
COMPRISE Sample App	Sample App which represents a user's smartphone project he/she wants to enrich with COMPRISE functionality	https://gitlab.inria.fr/comprise/comprise_sample_app

The COMPRISE Client Libraries consists of multiple sub-libraries, including both ready-to-market and existing technologies (Operating Branch) and technology developed within the course of the project (Training Branch). Both branches are available on NPM to be installed in the Angular Apps.

For the Operating Branch, all GitLab resources are theoretically available, but currently limited to the consortium. These components are developed to support the Proof-of-Concept for the Training branch components and are important for the project, but not "new" to the market. The focus clearly lies on the Training Branch components. Additionally, the consortium wants to keep the resources available for the external user small and only present the relevant, most important resources. Of course, access can be requested and will be granted, if desired.

COMPRISE Client Libraries Components (Operating Branch)		
Component name	**Link to GitLab**	**Link to NPM**
Speech-To-Text	https://gitlab.inria.fr/comprise/sdk_components/comprise_speech_to_text	https://www.npmjs.com/package/comprise_speech_to_text
Machine Translation	https://gitlab.inria.fr/comprise/sdk_components/comprise_machine_translation	https://www.npmjs.com/package/comprise_text_to_speech
Spoken Language Understanding	https://gitlab.inria.fr/comprise/comprise_natural_language_understanding	https://www.npmjs.com/package/comprise_natural_language_understanding
Dialog Management	https://gitlab.inria.fr/comprise/sdk_components/comprise_natural_language_generation	https://www.npmjs.com/package/comprise_natural_language_generation
Text-To-Speech	https://gitlab.inria.fr/comprise/comprise_natural_language_generation	https://www.npmjs.com/package/comprise_text_to_speech

As part of the research activities, all Training Branch components are completely open source, publicly available and allow any kind of contribution.

COMPRISE Client Libraries Components (Training Branch)		
Component name	**Link to GitLab**	**Link to NPM**
Privacy-Driven Text Transformation	https://gitlab.inria.fr/comprise/text_transformer	https://www.npmjs.com/package/comprise_privacy_driven_text_transformation
Privacy-Driven Speech Transformation	https://gitlab.inria.fr/comprise/voice_transformation	https://www.npmjs.com/package/comprise_privacy_driven_speech_transformation
Cloud Platform API	https://gitlab.inria.fr/comprise/sdk_components/comprise_cloud_platform_api	https://www.npmjs.com/package/comprise_cloud_platform_api

5 Outlook

Deliverable D4.5 marks the final version of the COMPRISE SDK and presents a software which is ready to be used by external stakeholders. Nevertheless, on the way there, the project partners, as well as external users had the chance to test and contribute to the creation process, which identified some potentials to further improve the COMPRISE SDK. As the project aims towards a successful exploitation, continuous testing, bug fixes and improvements will continue. Some of the tasks below will continue to be addressed:

- Make TTS models downloadable dynamically the same way as realized for STT.
- Moving libraries back from COMPRISE Personal Server to native representations to make the COMPRISE Client Libraries work more “on device”, as initially planned. This work is already in closer investigation for STT and TTS.
- Replace the last few external components such as Machine Translation by a version running on the COMPRISE Personal Server to further preserve privacy.
- Extend the functionality of the COMPRISE Personal Server in a way that App end-users can also setup their own COMPRISE Personal Server in a quick and easy way, while keeping the developers’ configuration.
- Improve the performance of the voice data processing chain.
- Resolve remaining integration issues for both Android and iOS deployment.
- Additional issues reported by users.

6 Conclusion

This deliverable described the final prototype of the COMPRISE SDK, consisting of the COMPRISE Personal Server, the COMPRISE Client Libraries, and the COMPRISE App Wizard.

According to the implementation plan in Deliverables D4.1 and Deliverable D4.3, the COMPRISE SDK has successfully reached all major integrations targets, with only some minor potential bugs to be resolved.

The main improvements compared to the architecture initially presented in Deliverable D4.1 and to the first architecture in Deliverable D4.3 was the introduction of the COMPRISE Personal Server, the dynamic way of receiving models and integrating updated functionality of Work Package 2 and Work Package 3, which enabled the COMPRISE SDK to use the project’s latest results. For all components, documentation has been created online. The documentation will be updated regularly, depending on future bug fixes and SDK improvements.