REPORT

Final system architecture and API specification

- project deliverable 3.2

Authors: Jarkko Kuusijärvi, Markku Kylänpää, Kimmo Ahola, Timo Laakko (VTT); Symeon Papadopoulos, Chrysanthi Iakovidou, Vasiliki Gkatziaki (CERTH); Barbara Guidi, Andrea Michienzi, Laura Ricci (UNIPI); Kristina Kapanova, Kevin Koidl (TCD); Tommi Meskanen, Valtteri Niemi (UH); Francesco D'Andria, Carlos Alberto Martin Edo, Ignacio Dominguez (ATOS); Daniel Bautista, Vanessa Clemente (WLI).

Confidentiality: Public

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825585.

HELIOS Final system architecture and API specification

Project name	**Grant agreement #**
HELIOS	825585
Authors Jarkko Kuusijärvi, Markku Kylänpää, Kimmo Ahola, Timo Laakko (VTT); Symeon Papadopoulos, Chrysanthi Iakovidou, Vasiliki Gkatziaki (CERTH); Barbara Guidi, Andrea Michienzi, Laura Ricci (UNIPI); Kristina Kapanova, Kevin Koidl (TCD); Tommi Meskanen, Valtteri Niemi (UH); Francesco D'Andria, Carlos Alberto Martin Edo, Ignacio Dominguez (ATOS); Daniel Bautista, Vanessa Clemente (WLI).	**Pages** 6+38
Reviewers Robin Ribback (STXT), Rolf Nyffenegger (STXT); Morgan Fredriksson (NG)	
Keywords System architecture, security, API specification, components, core	**Deliverable identification** D3.2

Summary

This deliverable presents the final version of the HELIOS system architecture specifications and the related API specifications. The document presents different architectural options for the overall HELIOS platform, which comprises the core, the extension modules to it and the applications utilising their functionalities. The architecture is generic enough to be implemented in various operating systems but the document refers to the Android operating system in some sections, as the project has selected and Android-first approach. In addition to the overall architecture, the document presents the currently planned core modules and extension modules and applications, presents the functionalities of each of them and provides API descriptions for them. This deliverable builds upon D3.1 "Draft system architecture and basic API specification".

Due to the iterative methodology followed in the project, the detailed architecture choices and the detailed API specifications will be updated according to inputs and feedback from the other work packages and tasks throughout the project. The APIs will be updated throughout the implementation of this architecture in the corresponding repositories of the project.

This document is public and is available at the project homepage for download.

Confidentiality	Public

25/06/2020

Editor
Jarkko Kuusijärvi

Project Coordinator's contact address
Ville Ollikainen, ville.ollikainen@vtt.fi, +358400 841116

Distribution
HELIOS project partners, subcontractors, the Project Officer, and HELIOS website.

Contents

List of Acronyms

Acronym	Description
ABAC	Attribute Based Access Control
AES	Advanced Encryption Standard, symmetric-key algorithm
API	Application Programming Interface
APK	Android application package
AR	Augmented reality
BLE	Bluetooth Low Energy
BPMN	Business Process Management Notation
CEN	Contextual Ego Network
DFRN	Distributed Friends & Relations Network
DOSN	Decentralized Online Social Network
E2E	End-to-end Encryption
GDPR	General Data Protection Regulation
HMAC	Hash-based message authentication code
HSN	Heterogeneous social network
HLS	HTTP Live Streaming
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IoT	Internet of Things
IPC	Interprocess communications
JSON	JavaScript Object Notation
NAN	Neighbor Awareness Networking
NAT	Network address translation
NFC	Near Field Communication
P2P	Peer-to-peer
REST	Representational state transfer
RPC	Remote procedure call
RSA	Rivest–Shamir–Adleman, public-key cryptosystem
RTMP	Real-Time Messaging Protocol
SDK	Software Development Kit
STUN	Session Traversal Utilities for NAT
TLS	Transport Layer Security
TURN	Traversal Using Relays around NAT
URL	Uniform Resource Locator
VoD	Video on Demand
VR	Virtual reality
WCAG	Web Content Accessibility Guidelines
WebRTC	Web Real-Time Communication
QoE	Quality of Experience

1. Introduction

Important notice

Due to the iterative methodology used in the project, this document presents the final architecture description for the HELIOS platform which will still be continuously updated according to inputs and feedback from the other work packages and tasks throughout the project.

After this D3.2 architecture specification, the module specifications and related API specifications will be developed further throughout the project and versioned accordingly in the project repositories, thus carrying out the iterative development process of the HELIOS project.

Therefore, this document specifies the overall architecture. The detailed module and application specifications will be updated as the development progresses and feedback from the other tasks implementing our vision will be fed back to the design process.

1.1 HELIOS definitions/terms used in this document

This section describes the overall definitions and terms used throughout this document; in the way the HELIOS project defines them. The order of the descriptions is based on the meaning and inter-relationships of the terms, i.e., not in alphabetical order.

Peer-to-peer (P2P) application

P2P is a decentralised communication model where each participant has the same capabilities and participants can start communication with other participants without intermediation by a third party. They have both client and server functionalities at the same time. In the context of HELIOS, a P2P application is referred to as an (Android) application running on the user's device and uses HELIOS core's Communication Manager APIs for exchanging messages and other content with the other peers. Later in the document, application, modules, or core component always refer to Android implementation, if not specifically mentioned otherwise.

HELIOS user/node/actor

A HELIOS user is a human user of the system. A HELIOS node, in general, is a device that has the HELIOS core installed is visible in the HELIOS network as a HELIOS node. An actor (see the below User profile definition for additional details) can be a person, a sensor, a social bot, an organisation, or other kind of agent, such as social network spiders or crawlers. Actor definitions and properties can be seen in details in D4.1, Section 4.4. These terms are tied closely together in this deliverable, normally a user refers to a human user of the system.

Trust

Trust is a complex concept, which considers several aspects of the agents' dynamics. Trust has several properties that are usually used to define models or to measure it. Some of them include propagative, non-transitive, asymmetric, and composable. Dynamics means that trust can change. Indeed, it can increase, decrease, or decay with time. In general, there are two approaches to update the trust value: event-driven, where all trust data are updated when an event happens, and time-driven, where trust is periodically updated. In Online/Offline Social Networks, trust has been studied

in depth, and an important property of trust is that it is propagative, which means that if a generic user A trusts B, and B trusts C, A can also trust C. However, trust is not transitive, which means that if A trusts B, and B trusts C, this does not imply that C trusts A. Trust is asymmetric, which means that a level of trust between two members is not the same. In HELIOS trust is an important concept in order to manage the decentralized social network of users. Indeed, a trust value is assigned to each relationship and it is used to evaluate the strength of a specific tie.

Context

Context is any information (e.g., **spatial**, **social**, and **temporal**) that characterises the situation in which each user can find themselves (egocentric view). A context may include a type and several attributes that are used to define and detect the context, and a state (active or inactive). Values of context attributes may be determined explicitly (e.g., a given static value) or implicitly (e.g., by an external context source, sensor value). In HELIOS, a context is associated with a layer of CEN.

Heterogeneous social network (HSN)

A graph is a collection of nodes and edges that represents specific relationships, where nodes correspond to users and edges are the connections between users. Although social networks were originally conceived as a model of capturing links and interactions among humans, it is nowadays clear that modern socio-technical networks may comprise a variety of actors. In HELIOS, networks may also include non-human actors like smart devices, sensors, social bots, or organisations corresponding to nodes. To model such a network, HELIOS implements the social overlay with a Heterogeneous Social Network (HSN) graph. The HSN is the union of each user's local view implemented by exploiting the Contextual Ego Network.

User profile

Each node (or a user if referring to humans) in the HELIOS network is associated and described by a set of attributes. These attributes may be associated with a HELIOS actor through the users' explicit input (e.g., name, age, gender, etc.). Other attributes may be inferred based on the information provided from sensors on the user's device (e.g., location, language, etc.) or extracted through implemented processes for mining the social ego-graph, analysing interaction patterns, or revealing content consumption preferences, to name a few. The collection of the attributes that characterise an actor are organised into a linked data schema (concepts and corresponding properties), and the final user profile will be stored as a machine-readable schema accessible through the HELIOS core. Access rights to different data that are stored in the user's profile will be granted based on each users' preferences and can be semi- or fully- automated based on the trust and context criteria. **Note:** The user profile of the core can be separated from the user profile defined in the contextual ego-network (CEN) (see below), as the profile used by core can include, e.g., user's privacy settings, encryption keys, etc. The profile information is shared throughout the components, though.

Emotional feedback/score

Related to the user (humans), an emotional feedback is defined as how the users response to a specific stimulus (e.g., analysing the reactions of the users to a specific content received using facial recognition). An emotional score, on the other hand, is calculated based on the analyses of the user's emotional statements between users (e.g., if one user harasses another, it is calculated based on the analysis, which is then evaluated and reflected by the probable decrease the score). Refer to the Neuro-behavioural classifier module for further info.

HELIOS Contextual Ego-Network (CEN)

In HELIOS, the social network of each user is represented by the Contextual Ego Network, implemented by exploiting the ego network social model. The ego network represents a structure built around the user, also known as the "ego", which contains their direct connections with its friends (alters) and the alter-alter connections. The ego network is an important social structure usually used to model the local social network view of a person. This represents an important structure in Decentralized Social Networks, usually used to implement the local view of a user, considering that the decentralization does not permit to have the knowledge of the whole social network.

Contextual Ego Network is a multi-layered social overlay consisting of a set of layers, where each layer represents a real-life context of the ego, and it is modelled by exploiting the ego network structure. Each layer is built upon the context of the ego, which defines the connections between the nodes in the layer. The connections may have different strengths based on the other context-social information of the network. The definition of the structure can be found from the project deliverable D4.1, and the formalisation in D4.2.

1.2 Connection establishment paradigms

The HELIOS project aims to provide a platform that achieves next generation privacy-aware social media connections between people and smart devices. This platform allows nodes (or normally users, i.e., humans) to communicate with each other directly peer-to-peer (P2P). In addition, it allows to form person-to-person connections with each other using either subscription or ad-hoc connection establishment methods. In the **subscription paradigm**, the user can deliberately add or subscribe to a specific person and / or interest. In the **ad-hoc paradigm**, the user's social network and connections can be formed based on the contextual properties and user profiles, respecting each users' privacy settings.

All the actions taken by individual users in different settings - independent of the method how the users have formed their connection initially - affect the overall trust between users. This allows the level of trust between users to increase or decrease, based on the connections and actions taken by the users in the context they find themselves in.

In all these cases, HELIOS uses *end-to-end encryption (E2E)* in the communications between the users (be that 1-to-1 or 1-to-many communications) to protect the users' data. HELIOS also allows users to select their preferences on to what degree the platform should form these connections and how much information is shared between other users in different situations.

1.3 Communication modes

The HELIOS project aims to implement support for both an **online communication mode** and an **offline communication mode**. Although the envisioned HELIOS architecture supports both modes, the online mode is the primary mode and the offline mode is an additional mode to be supported. In the online mode, also referred to as *HELIOS Online P2P*, the user has connectivity to the Internet and therefore can communicate with any user on the Internet. In the offline mode (*HELIOS Local P2P)*, the user only communicates with the people that are nearby and there is no connectivity to the Internet. This *HELIOS Local P2P* mode would be very beneficial in circumstances, where there is no connectivity to the Internet for any reason, or the users wishes to communicate with people that are near (in close proximity) to them and do not want to relay messages via the Internet.

Figure 1 depicts the two main modes discussed above. The colouring of the edges/communications between the users illustrate possible communication paths, i.e., E2E communication paths. For example, on the right-hand side, there are three separate communication paths (green, red, and blue edges) that illustrate different connections between the users. The main mode is on the left-hand side, where people can communicate via the Internet on secure connections between the other users (the messages are encrypted end-to-end). On the right-hand side, the *HELIOS Local P2P* mode, the people communicate with each other in close proximity, without Internet connectivity. The connections in the *HELIOS Local P2P* can be achieved via Bluetooth or Wi-Fi connections (e.g., Wi-Fi Direct[1]).

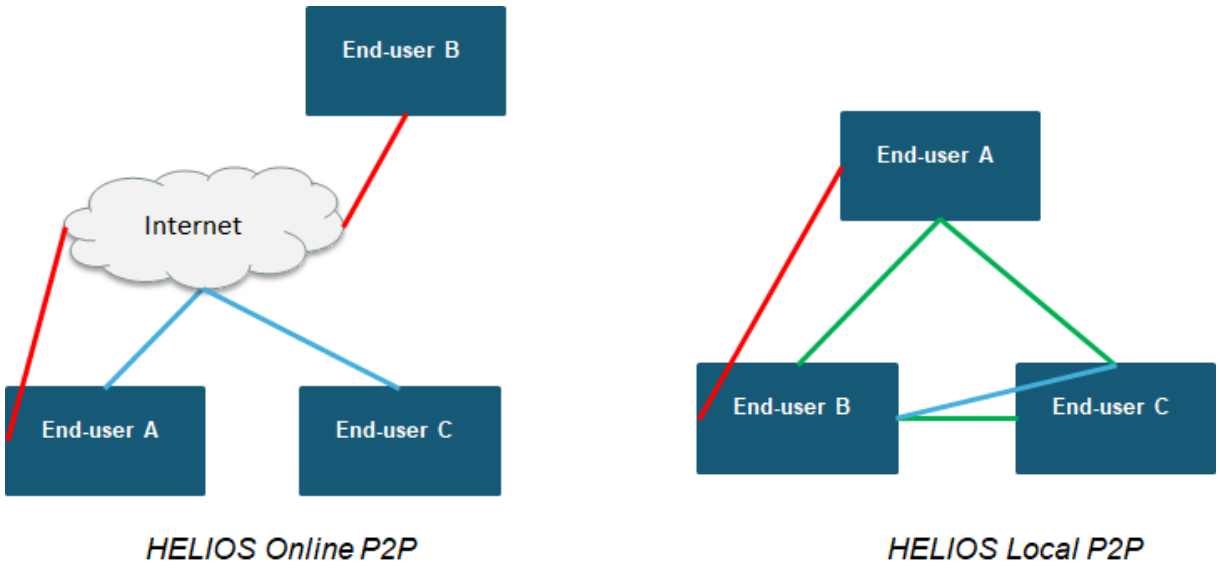

Figure 1. Main HELIOS communication modes.

In addition to those mentioned above, an additional mode has been defined, called hybrid mode. This mode is one that the HELIOS project envisions but is only considered to be supported in a later stage after the main modes have been implemented. In this mode, a user can enable other users to relay messages to HELIOS users on the Internet while they themselves are offline without Internet connectivity. This communication mode would be very beneficial in cases, where messages would have to be relayed via offline HELIOS users to other HELIOS users on the Internet, such as in the case of a natural disaster.

[1] https://www.wi-fi.org/discover-wi-fi/wi-fi-aware

2. Background

2.1 Decentralized Online Social Networks

A Decentralized Online Social Network[2] (DOSN) is an Online Social Network implemented in a distributed manner by exploiting decentralized platforms, such as a network of trusted servers, P2P systems or an opportunistic network.

An Online Social Network (and DOSNs) provides social services. The principal service is to allow communication between users. Indeed, in an Online Social Network (and in a DOSN) a social relationship is the permission to communicate and to share information, given by a user to another. Once communication is allowed, a communication channel is used to pass all private communications and data.
The following communication protocols will be examined while building the HELIOS Communication Manager (see Section 4.3.1). The Communication Manager takes care of the overall essential communication-related tasks in the HELIOS platform.

One of the main examples of DOSNs is Fediverse[3], in which several existing systems, like Diaspora[4] and Mastodon[5] are joint. Fediverse is a group of federated servers that are used for web publishing and file hosting. The systems in Fediverse run on servers that support one or more communication protocols and follow an Open Standard.

The communication protocols, available in Fediverse, are:

- **ActivityPub**[6]. ActivityPub is an official W3C[7] recommended standard protocol, which provides a client to server API for creating, updating, and deleting content, and a federated server to server API for notifications and subscribing to content.
- **DFRN**. DFRN is the acronym of “the Distributed Friends & Relations Network” protocol. It provides the communication basis for a decentralised social network - where cooperating servers share information on your behalf while operating in a web of trust relationships you control.
- **Diaspora Federation Protocol**. This is the protocol used by Diaspora. It allows four distinct platforms to all communicate: Diaspora, Friendica, Hubzilla, and SocialHome.
- **OStatus.** OStatus is used in federated social systems to allow users on one website to send and receive status updates with users on another website. In detail, it is a suite of protocols for distributed status updates or microblogging.
- **Zot & Zot/6.** It is an evolution of (and a simplification of) many concepts of the DFRN protocol. It is a communications protocol for social communications on the web, which provides decentralised communications and identity management.

A DOSN has a few disadvantages relative to the centralised network: latency issues, platform requirements, data availability, no central authority to privacy and security issues (spam, malware, etc.),

[2] “*Decentralized Online Social Networks*”. Handbook of Social Network Technologies and Applications. 2010. Datta, Anwitaman and Buchegger, Sonja and Vu, Le-Hung and Strufe, Thorsten and Rzadca, Krzysztof.
[3] https://fediverse.party/
[4] https://diasporafoundation.org/
[5] https://joinmastodon.org/
[6] https://www.w3.org/TR/activitypub/#social-web-working-group
[7] https://www.w3.org/

distributed authentication and authorisation. The HELIOS project will address these issues in the HELIOS platform components to minimise the disadvantages identified as much as possible.

2.2 Peer-to-peer libraries

There are several peer-to-peer (P2P) protocols and implementations to consider in the development of the P2P messaging module. In detail, the following were considered:

- **Scuttlebutt**[8]: a peer-to-peer protocol developed mainly in JavaScript that allows developers building decentralized applications that work well offline. Key features of scuttlebutt are:
  - ***Identities***: users of scuttlebutt need an identity to participate in the network. A Scuttlebutt Identity comprises of the secret and public key (an Ed25519[9] key pair)
  - ***Pubs*** are publicly accessible Scuttlebut peers. They function as a gathering point for new users to find other existing peers.
  - ***Private Messages*** are encrypted (end-to-end encryption by default) and only specified recipients can read them.
  - ***Feed(s)*** is a list of messages posted by a Scuttlebutt Identity
  - ***Following*** an optional function that allows users to follow the feeds of other users/scuttlebutt identities.
  - ***Blob*** referred to Scuttlebutt messages stored by peers and containing binary data.
  - ***Peer discovery*** over the local network, through invite codes and pubs.
- **DAT**[10]: a peer-to-peer protocol developed in Javascript for sharing data which makes changes in data transparent. The main goal of DAT is to allow users to store, track and share data over time. Key features of DAT are:
  - file transfers over an **encrypted connection**,
  - **peer discovery** with the use of a distributed hash table (Kademlia algorithm),
  - data exchange with the use of **Append-Only Logs** that let users download part of the log (Merkle Distributed Acyclic Graphs). The file system abstraction is built on top of **two Append-Only Logs**, one for raw content and one containing metadata related to the folder structure.
- **IPFS**[11]: a peer-to-peer hypermedia protocol mainly in Go and JavaScript for storing and sharing data and it uses content-addressing to uniquely identify files. Key features of IPFS are:
  - **peer discovery** by using a distributed hash table
  - **automatic NAT detection** to determine reachable nodes
  - offers three **gateways for HTTP** within IPFS: 1) path-based, 2) subdomain-based and 3) through DNSLink
  - uses **Bitswap protocol** for data exchange. The main purpose of Bitswap is to request and send blocks to other peers in the network.
  - **transport layer security** (TLS) by default
  - **EthDNS** allows access to data in the Ethereum Name Service from DNS without running an Ethereum client.
- **libp2p**[12]**:** a modular peer-to-peer networking stack available in multiple languages.
  - Several well-known P2P frameworks are using **libp2p** (e.g., **IPFS**)
  - **Implementations in multiple languages**, interoperability is good, reasonably large user base, actively developed and maintained.

[8] https://ssbc.github.io/scuttlebutt-protocol-guide/
[9] https://en.wikipedia.org/wiki/EdDSA#Ed25519
[10] https://datprotocol.github.io/how-dat-works/
[11] https://ipfs.io/
[12] https://libp2p.io/

- **Permissive license**, mostly MIT
- **Peer discovery** by using a distributed hash table.
- **Is transport agnostic**, so multiple transport protocols can be used like TCP/UDP and Websockets. Extendable built in transport security.
- **Built-in protocol encapsulation and negotiation** (like identify, ping, secio, circuit relay, kad-dht) and multiplexing streams, built-in protocol encapsulation and negotiation.
- **NAT traversal** using Circuit Relay protocol, identify protocol, UPnP or nat-pmp.
- **Every peer** has a **unique cryptographically** constructed peer id.
- **Every peer** can be **authenticated** and reached directly by using the peer id alone.
- **All** peers also **reachable** via explicit addresses **communication** between peers can be easily authenticated and encrypted (secio protocol).

- **BRAMBLE**[13]: an Android first protocol stack implemented in Java. BRAMBLE supports five protocols in total:
  - ***Bramble Transport Protocol (BTP)*** which is a transport layer security protocol suitable for delay-tolerant networks. It offers a time-based key management protocol and a wire protocol for securely carrying streams of data. BTP leverages Tor as underlying transport to conceal identities, achieve unlinkability and unobservability.
  - ***Bramble Synchronisation Protocol (BSP)*** is designed to synchronise data among peers in delay-tolerant networks. The protocol organises data into groups with independent synchronisation scopes that contain message graphs with immutable messages.
  - ***Bramble Rendezvous Protocol (BRP)*** which is a discovery protocol for peer-to-peer networks and uses Tor hidden service protocol as its transport. It enables users who have exchanged public keys to get connected.
  - ***Bramble QR Code Protocol (BQP)***, a key management protocol that establishes a shared secret key between two peers.
  - ***Bramble Handshake Protocol (BHP)***, a key management protocol that enables peers to establish ephemeral shared secret keys.

3. Requirements

This section describes the functional and non-functional requirements for the overall HELIOS system architecture, in the form of security & privacy and other technical requirements. These requirements address many points raised in the technical requirements analysis in T2.5 (the final requirements will be published in D2.9 “Technical Requirements Gathering”, M26). The requirements produced by T2.5 take into account the use cases planned in WP2 (See D2.1 “Initial Concept Report”, M6), envisioned by artists during the initial design fiction phase. This version includes the current technical requirement layers identified in T2.5, taken as a snapshot for creating the architecture plan and APIs.

Requirements in technical requirements task T2.5 are divided into Vertical and Horizontal layers, depicted in **Figure 2**.

- **Vertical layers (V-layer)**: Describes the requirement layers that are present in most of the use cases, thus affect nearly every use case.

[13] https://code.briarproject.org/briar/briar-spec/-/tree/master/

- **Horizontal layers (H-layer)**: Describes the requirement layers that are present in some of the use cases, not necessarily in everyone.

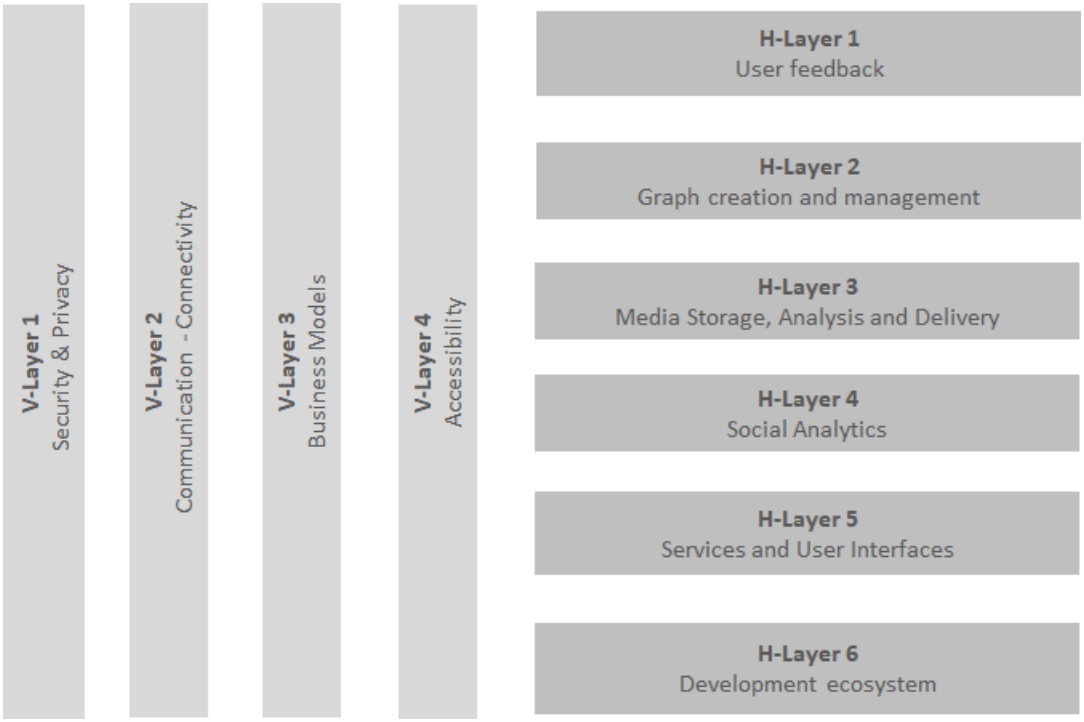

Figure 2. HELIOS requirement layers.

The following sub-sections summarise the current set of requirements (that are subject to change during the course of the project).

Not all layers might be needed for the use cases defined in the HELIOS project and might not be reflected in the core architecture. Therefore, the implementation of the individual requirements will be prioritised according to the agile planning process.

3.1 Security and privacy requirements

In order to provide a privacy-aware HELIOS platform, the following requirements need to be addressed in terms of security and privacy. These requirements present the **Vertical layer 1 (V-Layer 1)** of the requirement layers. This is the most important layer in terms of user's privacy and security.

- Methods to allow encryption of every piece of content the user introduces to the platform. This includes both data stored in the personal data storage and data sent to other users. The methods to generate keys for encryption/decryption/signing/signature verification are also needed.
- Methods for end-to-end encryption, both for one-to-one communication and one-to-many communication.
- Methods to distribute public keys between the users. This can be done either face-to-face or using some distributed database. In the latter case, there must be means to verify the authenticity of these public keys. E.g., Blockchain technologies can be used to distribute keys and smart contracts to check validity of signatures against public keys. One method to use blockchain is that every user writes their certificate that includes their ID, their public key and their signature

of ID and public key to the blockchain. This links the user ID and public key together and proves that the user knows the corresponding private key for signing. It is important to detect efficient methods for re-keying of data, for instance, when group communication is defined. User groups in DOSNs are characterised by a high level of churn, so keys have to be changed very frequently to guarantee back and forward secrecy. Existing methods for efficient re-keying present in the literature will be investigated.

- Methods for authenticating users.
- For a user, methods for authenticating messages that the user receives.
- Methods for protecting integrity of contents. The methods to generate keys for integrity protection/verification are also needed.
- Methods to retrieve and manage public keys of other users.
- Management of keys (both secret and public keys) the user has.
- Methods to control access to the user's device.
- Methods to control access to the user's profile information.
- Methods to control access to the user's data.
- Methods to maintain the anonymity of a peer.
- Methods to store control data such that the user cannot modify it. This is needed, for example, to store a counter about how many reads a post has. This is about securing the integrity of control data between two instances of the platform.
- Methods to compare the characteristics of user data between two HELIOS instances, in a privacy preserving manner.

To meet the above requirements, Transport layer security (TLS/HTTPS) can be used when delivering messages between users. The message content could be encrypted with keys matching to the delivery context/social network.

GDPR and legal/ethical compliance

Compliance with legal requirements emanate from the desire to apply the principle of data protection by design and by default from the GDPR to HELIOS Platform regarding protection of personal data. The following technical and organisational measures shall be addressed to evaluate and mitigate possible risks in the processing and to avoid data breaches.

- Pseudonymisation and encryption of personal data.
- Ensure confidentiality, availability, resilience of processing.
- Restore availability and access to personal data in case of physical/technical incident. Reviewing/auditing system functions for effectiveness and fit for purpose.
- Ensure a process for testing, accessing and evaluating the effectiveness of these technical and organisational measures for ensuring security regarding, for instance, accidental or unlawful destruction, loss, alteration, unauthorised disclosure or access to personal data transmitted or stored.
- ***Note****:* These requirements must be considered as they apply to different modules produced, depending on the data that is handled, etc.

Regarding individual rights to be exercised by data subjects (identifiable users of the platform), the different modules in the platform and the HELIOS core shall grant the technical exercise of the following GDPR rights: Right of access (Art. 15); Right to rectification (Art. 16); Right to erasure (Art. 17), remove

all data about a user who requests; Right to restriction of processing (Art. 18); Right to data portability (Art. 20); Right to object to processing of personal data (Art. 21); Right to withdraw consent, without affecting the lawfulness of processing before the withdrawal is exercised; Right to lodge a complaint before a supervisory authority (Art. 77) and the Right not to be subjected to automated-decision making including profiling (Art. 22). For more information see D2.3 legal, ethical and gender requirements.

3.2 Technical requirements

The following presents initial requirements for Vertical layers 2 to 4.

V-Layer 2: Communication - Connectivity

- QoS-related support (to be determined based on availability, average, maximum delay, throughput, depending on the used technology).
- Collecting automatic feedback from message recipients.
- Support P2P communications network for mobile devices that can deliver messages over legacy IP network (referring to normal networks used, e.g., 3G/4G/5G, Wi-Fi, etc.).
- Support P2P communications network for mobile devices, which can operate over legacy IP networks in such a way that if one of the phones gets disconnected from the network, it will receive a message when connecting again.
- Support analytical data over content usage, e.g., message delivery.
- Support operation of HELIOS without any centralized server so that it is free of geographical constraints.
- HELIOS supports wireless ad-hoc connectivity like Wi-Fi, Bluetooth, and P2P local connectivity based on before mentioned technologies.
- Support Ad-hoc Network Creation and Configuration. From a technological point of view, the ad-hoc aspect relies on being non-intrusive and requiring no user interventions. The temporal ad-hoc social networks are created based on users' social context and underlying wireless technologies. A key feature of this ad-hoc connectivity approach is to ensure there is a sense of proximity.
- Connectivity core, also referred to as the Communication manager, will be the main component for providing an efficient platform and a protocol agnostic way of communication for different entities.
- Using the Android operating system, different components in HELIOS platform can be connected using either Intents or lower level IPC mechanisms and can be orchestrated to work together with extended functionality. If applications are built from multiple components, it requires that security issues are considered (see security and privacy requirements).
- Support storing the discovery information of peers in a context in the core and/or request information from HELIOS Rendezvous nodes. Rendezvous nodes are thought in case the user does not have a public IP address that can be reached for direct communications.

V-Layer 3: Business Models

- Support Rewarding Business Model (e.g., crowdsourcing) and other business models. D2.5 "Commercial Exploitation Requirements Gathering" presents more requirements on this layer, but as it is a confidential deliverable; they are not listed here in detail.

V-Layer 4: Accessibility

The accessibility requirements will be based on the four high-level principles presented in Web Content Accessibility Guidelines (WCAG) 2.1[14], as follows:

- Perceivable - Information and user interface components must be presentable to users in ways they can perceive.
- Operable - User interface components and navigation must be operable.
- Understandable - Information and the operation of user interface must be understandable.
- Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies (e.g., screen readers).

The following presents initial requirements for Horizontal layers 1 to 6.

H-Layer 1: User feedback

- Sensors and methods to characterise the behaviour of the users, e.g. with text sentiment analysis, image analysis, and voice analysis.
- Restrictions in data recollection derived from security, legal and data-minimisation guides.
- Model to characterise the relationship between an ego and an alter in terms of emotion and behaviour.
- Emotional score to feed trust management.
- Model to characterise a content (e.g., Cross Reality (XR) contents) to give emotional feedback.
- Pipeline and storage of the data.

H-Layer 2: Graph creation and management

- Explicit user input.
- Ad-hoc graph creation (node, edges) based on sensors and other devices.
- Index user profile and contact info.
- Group management functionalities.
- Visibility-management of shared personal data.
- Provide ego network links (ego-alters, alters-alters)
- Time-varying scores (trust score, emotional score, etc.)
- Sensor-related data – location (latitude, longitude), specific time/time zone, proximity, and others.

H-Layer 3: Media storage, analysis and delivery

The System needs to be able to support/provide:

- Store multi-media files.
- Management of live and Video on Demand (VoD) media content.
- Encoding capabilities (including on the fly encoding for live performance).
- (Premium) Media distribution capabilities (VoD and live).
- Adaptive streaming capabilities (e.g., Dynamic Adaptive Streaming over HTTP).

[14] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=111%2C131#principle4

- Different set of qualities to enable premium business models.
- Preservation of metadata and associated contents (e.g., access services).
- Video distribution using P2P infrastructure (P2P).
- (Device-Device P2P) Live video transmission and teleconference directly between devices (without or with Internet connectivity).

H-Layer 4: Social Analytics

- Access to communication primitives, in order to develop protocols able to communicate among known nodes.
- Access to storage of the results in a multilayer graph structure.
- Discover and make available the current context.
- Graph mining and knowledge discovery algorithms.
- Detect a contextual network (get same context neighbours).
- Extract trust scores between linked nodes.
- Aggregate media content scores along facets.
- Build anonymously user profiles (by analysing user inputs)

H-Layer 5: Services and User Interfaces

- Accommodate user and group communications and recommendations.
- Context-content interaction-based recommendations.
- Enable setting and updating profile information.
- Group-based push notification.
- Support of Augmented Reality (AR)/Virtual Reality (VR) interfaces.
- P2P publishing supporting prosumer production and premium content management.

H-Layer 6: Development ecosystem

- Scalable and extensible architecture.
- Support mobile application configuration by non-experts (also find and update modules for supporting additional features).
- Support accessibility through UI design.
- Interoperability between different operating systems through core modules.
- Interface and API definitions for core and extension modules.
- Documentation of the HELIOS architecture and the relevant APIs.

3.3 Mapping requirements to HELIOS Objectives

The following **Table 1** presents the mapping of HELIOS Objectives (See Annex 1 for reference) to the technical requirement layers presented before. **Table 1** omits objectives that are not in the scope of the technical architecture, i.e., are general objectives or business objectives. The omitted objectives in this table are: **OBJ3** (Validate the platform with real users that share common interests in communicating with each other) and **OBJ4** (HELIOS will be based on Open Source principles). OBJ3 is validated in WP7 and OBJ4 is validated by using Open Source components and by making the source code of the core components publicly available.

Table 1 presents the most important layers involved for fulfilling the objective. For instance, the vertical layers should be present in any objective, but the most relevant ones are listed with each objective.

Table 1. Requirement layer mapping with HELIOS objectives

Objective ID	Requirement layer	Comments
OBJ1	V1, V2, H2	
OBJ2	V1, V2, H2	
OBJ5	V1, V2, V4	
OBJ6	H6	
OBJ7	V1, V2, V3, V4, V5, H3, H6	
OBJ8	V1, V2, H2, H3, H4	
OBJ9	V1, V2, H1, H2, H3, H4	
OBJ10	V1	
OBJ11	H5	Also, use cases from the design fiction in WP2, created by artists.
OBJ12	V1, H3, H5	
OBJ13	V1, V2	
OBJ14	V1, V2, H1	
OBJ15	V4	

4. HELIOS architecture overview

This section describes the high-level HELIOS architecture, the functional specifications of the main HELIOS components and their interfaces. The main components discussed in this section are:

- **HELIOS core,**
- **extension modules, and**
- **the applications**.

The architecture is based on the different Vertical and Horizontal layer requirements. The individual core components fulfil the more detailed requirements together with the foreseen extension modules and/or

applications. The applications (WP5) will fulfil the requirements extracted from the use case descriptions, user stories and user interface designs from WP2, D2.8 (and earlier versions of it during the development cycles). As such, the HELIOS architecture takes into account the overall requirements found in the layered requirements described in Section 3.

HELIOS is following an Android-first strategy throughout the project. The HELIOS platform requires the user to install the individual applications, extension modules and the core available from the Google Play store. The HELIOS core includes a basic GUI application (Core GUI), so that the user can only install the core and use the basic functionalities of the platform. These functionalities may include, e.g., user account creation, privacy settings, basic chat and media content sharing.

4.1 HELIOS architectural options

This section presents options for the logical architecture of the HELIOS platform. This includes the inter-relationships of the HELIOS core, the HELIOS extension modules and the HELIOS application(s). The architecture is generic enough to be implemented for multiple platforms but in the scope of this project and this deliverable, the Android Operating System is the target platform.

Permissions to communicate between HELIOS core, extension modules and applications must be established during installation and/or at runtime. All these components can handle users' data and must be trusted by the users to handle their data securely and appropriately.

In the case of HELIOS core, it is provided by the HELIOS project (initially within the project and after the project by the HELIOS community), and it can be trusted based on the open source software code, testing, validation, and the documents detailing the overall architecture and implementations. The details for the community and options related to open source are not considered in this deliverable. Fingerprint of the application signing for the core package can be provided for the developers in order to verify the package and the connection to the authentic core (in case of IPC connection).

The HELIOS applications and possibly extension modules can be developed by anyone, in case of developing extension modules, they can be suggested to the HELIOS community. When installing and taking an application/extension module into use, the users must decide whether they trust the application/extension module, or not. When taking the applications/extension modules into use, the HELIOS platform can ask the users whether they want to allow that specific application/module to access and handle user's data on a specific interface/permission. For requirements and discussion on the privacy issues related to this, see Section 3.1.

- If the answer is **YES**, that specific application/module can do anything with the data specified in that interface/permission.
- If the answer is **NO**, the user can still use the HELIOS platform and other features, just not the specific feature/features that application/extension module is providing.

All the presented options in the next subsections have their pros and cons, especially when considering the Android implementation of the architecture. The essential pros and cons or security considerations of each option are explained with the corresponding description. The option 1 will be used in the first implementations, while option 2 will be used at later stages.

4.1.1 Option 1 - Core, extension modules and applications in one package

Figure 3 depicts the option where an application developer includes/links HELIOS core and extension modules to the application directly. This can lead to situations where users have different core installed more easily than the latter options. The benefit of this option is, though, that the messaging between core and the application occur directly inside one package/process. As HELIOS is considering the core to handle only one user (user profile and account), the user will have to create new profiles for separate HELIOS applications using this option. Nevertheless, this option is aimed more at having an application providing multiple features as one and/or user wanting to have a more secure version of HELIOS core, this is not a crucial downside.

Using this option, the HELIOS core is implemented as an SDK library. This library can be hosted, e.g., on a NEXUS[15] repository for automatic management and distribution, or manually distributed to the applications developers via GitHub. The important thing here is to develop a good versioning strategy. One possibility is to use, for instance, Semantic versioning[16].

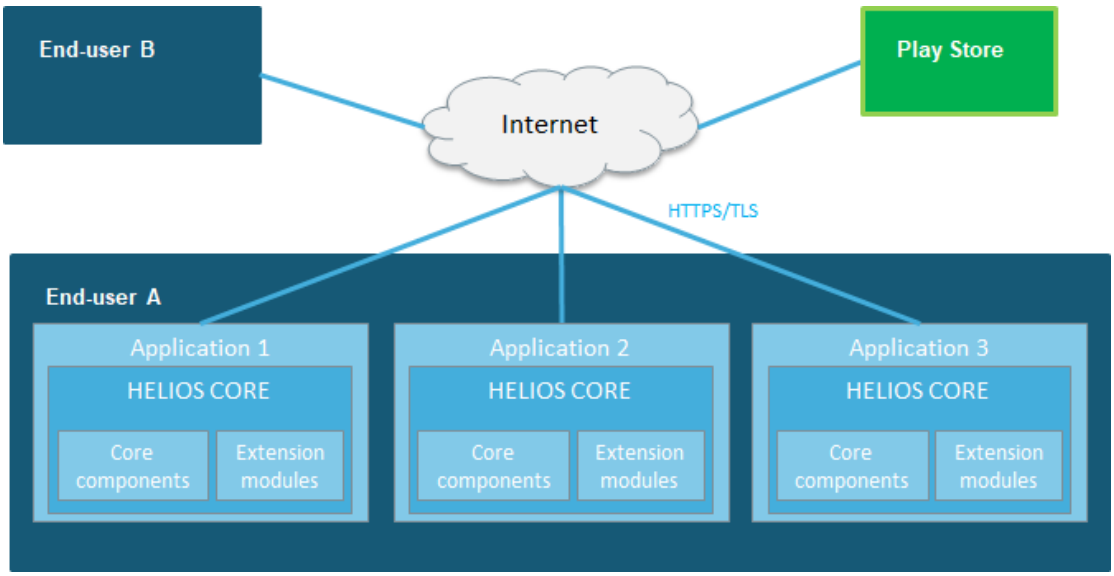

Figure 3. HELIOS components and applications all in one package.

Note for security: In this option, all the HELIOS modules are running in the same context/process, thus the user only must trust one application developer with their data. The user's data is only handled within that application.

4.1.2 Option 2 - Separate core and extension modules from the applications

Figure 4 depicts an option, where HELIOS applications communicate with the HELIOS core (and extension modules) inside the user's device. Thus, applications, core and extensions are separated and can be developed and published by separate developers, while keeping the core and extension modules in one package.

[15] https://www.sonatype.com/nexus-repository-sonatype

[16] https://semver.org/

The “core package” will also be an Android application, and in this scenario, there are two possibilities for the core application:

1. The core application will contain a screen that will serve as a launcher for the other applications. It will also contain all the extension modules.
2. The core application will only contain the extension modules, but no default activity. This means that while this application can contain UI elements through some of the extension modules (which can be called from the other applications, for example a shared login module), the application will not have an icon, or a shortcut, on the home screen. This can be achieved by not including a launcher activity in the manifest.

The “core package” will be updated by the HELIOS community (during the project, the HELIOS project), allowing new extension modules to be developed into the open source repository. The extension modules can be included into the HELIOS extension modules, once they have been thoroughly reviewed, tested, and validated.

For this approach to work, each HELIOS application should check if the HELIOS core application is installed, and if not, redirect the user to the Google Play store to download and install it.
Note: In Android, there is also an option to create separate packages for the extension modules, sign the packages with the same signature and have the two packages treated as one app[17].

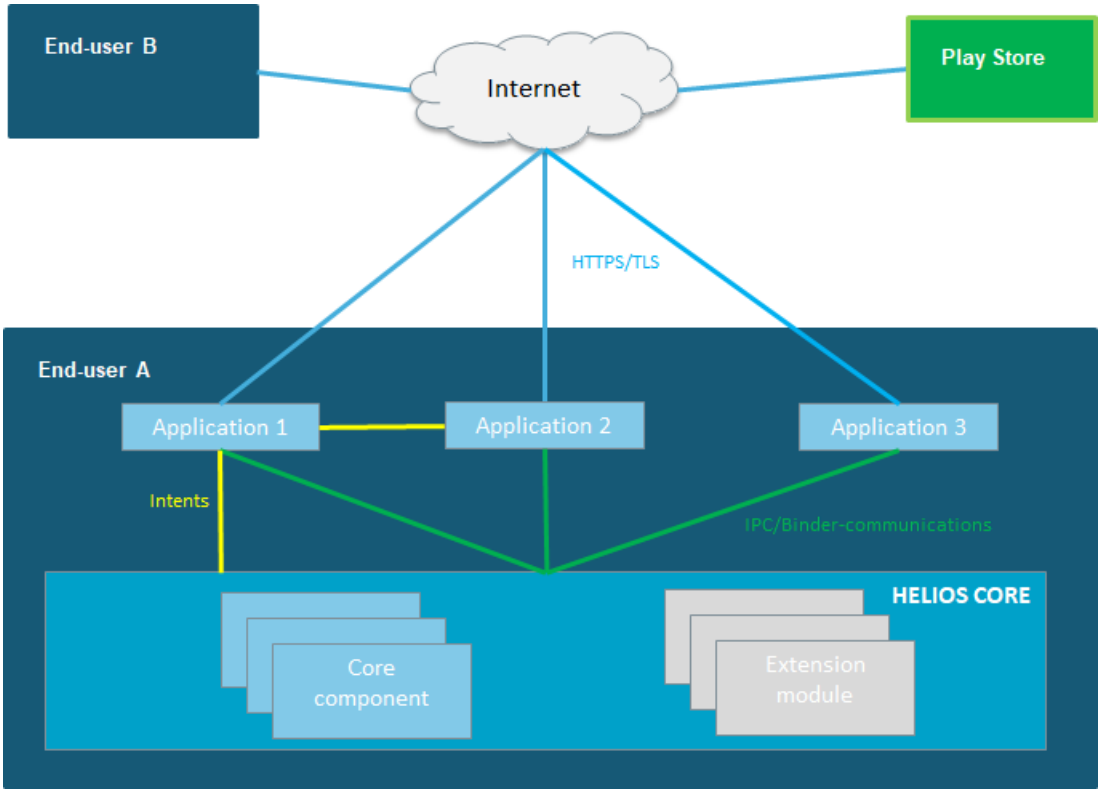

Figure 4. HELIOS core and extension modules separate from applications.

[17] https://developer.android.com/guide/topics/permissions/defining.html#userid

Naturally, application developers can include other libraries outside HELIOS to the application, and/or consume/provide information to other applications as well. The extension modules included into the core are more essential to the functionality of the HELIOS platform and the applications.. Of course, some core components can be used also individually outside of the HELIOS platform.

Note for security: This option has more implications to the security design of the HELIOS platform, since in this option the data must travel from the HELIOS core to the application (GUI and logic). This interface can be restricted by permissions, i.e., the user gives specific permission(s) for the application to access their data inside the HELIOS core.

4.1.3 Communication between applications and the core

As shown in Figure 4, there are two ways that an application can exchange data with the core: via 1) Intents and 2) via IPC/Binder-communications. The selection between these choices depends on what the application wants to produce to the core or consume from the core. Both of these can be supported at the same time, to allow flexibility in terms of integrating/communicating with the HELIOS applications (or core). Both of these options require the HELIOS core package to be installed, and the application should check whether it is installed or not and point the user to install the package if it is not installed (redirect the user to Google Play store in case of Android).

Intents

Intent handling can be included into the HELIOS application side or in the core GUI side, depending on the information/data that is to be shared. With Intents, other applications can share simple data (such as text, URL, or binary data, e.g., pictures or media files) directly, or share files through Android FileProvider (see https://developer.android.com/training/sharing/send). Core can require permissions in order to send content to it, or allow it without them.

The benefit is that this does not require the inclusion of a library into the application or implementing the IPC interface in any way, while other applications can still share data to the core and HELIOS application.

The downside is that we cannot share user data (besides pictures or files the user wants to share) through this method, since it would allow accessing private data of the user without proper privacy control.

Security and validation of the input is crucial when supporting Intents. Usually sending is one-way using Intents, although some result can be provided, however, for bi-directional communication with the core, the IPC/Binder-communication interface is preferred.

IPC/Binder-communications

IPC communications are handled either with a full Android Interface Definition Language (AIDL[18]) interface, or by using Messenger[19]. If application and core reside in the same process, they can also use a simple Binder (a service), but in the case of third party application developers, AIDL/Messenger is required.

[18] https://developer.android.com/guide/components/aidl

[19] https://developer.android.com/reference/android/os/Messenger

AIDL supports authentication of the client to the service, so it will be the preferred way to implement this interface. AIDL as a bound service connects directly to the client, without using broadcasts or Intents.

The data of individual HELIOS applications will have to be separated on the core service side, in order to properly handle user's internal application data. Another way to strongly separate the user's data in separate applications would be to allow only one HELIOS application to bind to the core service at a time. The downside of this option is that other applications would not receive messages when another application is active, so this option is not considered in the beginning.

The benefits of full IPC communications are that we can authenticate the other party with signatures, protect the interface with permissions (user accepts usage) and it is secure compared to sending content with a simple Intent (e.g., in case of text). The downside is that the application developer will have to import the AIDL files so that the Android SDK can generate the interface from it, or simply include the HELIOS library providing this.

The data packages between different layers are presented in Figure 5: 1) The application level will process its own data and provide an App-ID in the first level. This message is sent to the core via the **IPC/binder-communications.** 2) The HELIOS core packs the message that is then handled by the **Communication Manager** component to include HELIOS core messaging data and possibly, to encrypt it, providing the necessary information without encryption to the **P2P Messaging** module. 3) The low-level module uses the protocol defined by it and determined by feature/service being used.

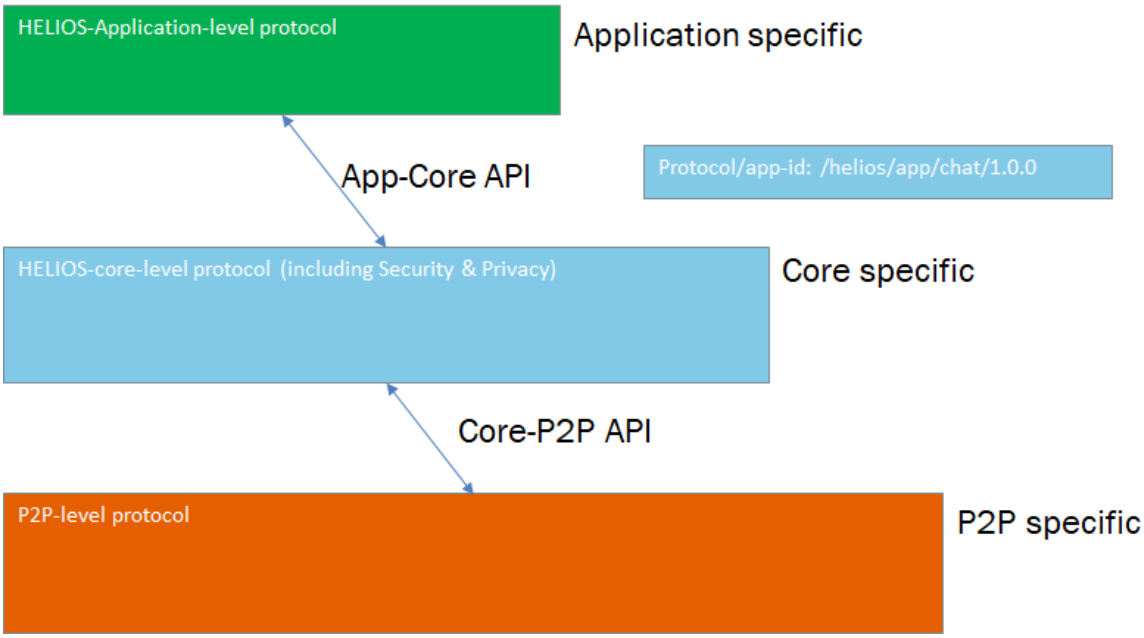

Figure 5. HELIOS API layers.

4.2 HELIOS logical architecture

The following section presents an option for the logical architecture of the HELIOS platform. It describes the components in terms of the previously explained Option 2 (Section 4.1.2). It should be noted that the same components can also be used in the same way using the other previously described options. The only difference is the access control/permission levels of the interfaces between the components.

The following descriptions also include the foreseen inter-dependencies of the HELIOS core, the HELIOS extension modules and the HELIOS applications. The architecture is general enough to be implemented for multiple platforms but in the scope of this project and this deliverable, the HELIOS partners are considering the Android Operating System as the first choice. The following also presents some of the foreseen core components, extension modules and applications.

The main purpose of this section is to give more details to the different components in the HELIOS platform. The functionality of selected components presented in the **Figure 6** will be explained in more detail in the Sections 4.3, 4.4 and 4.5.

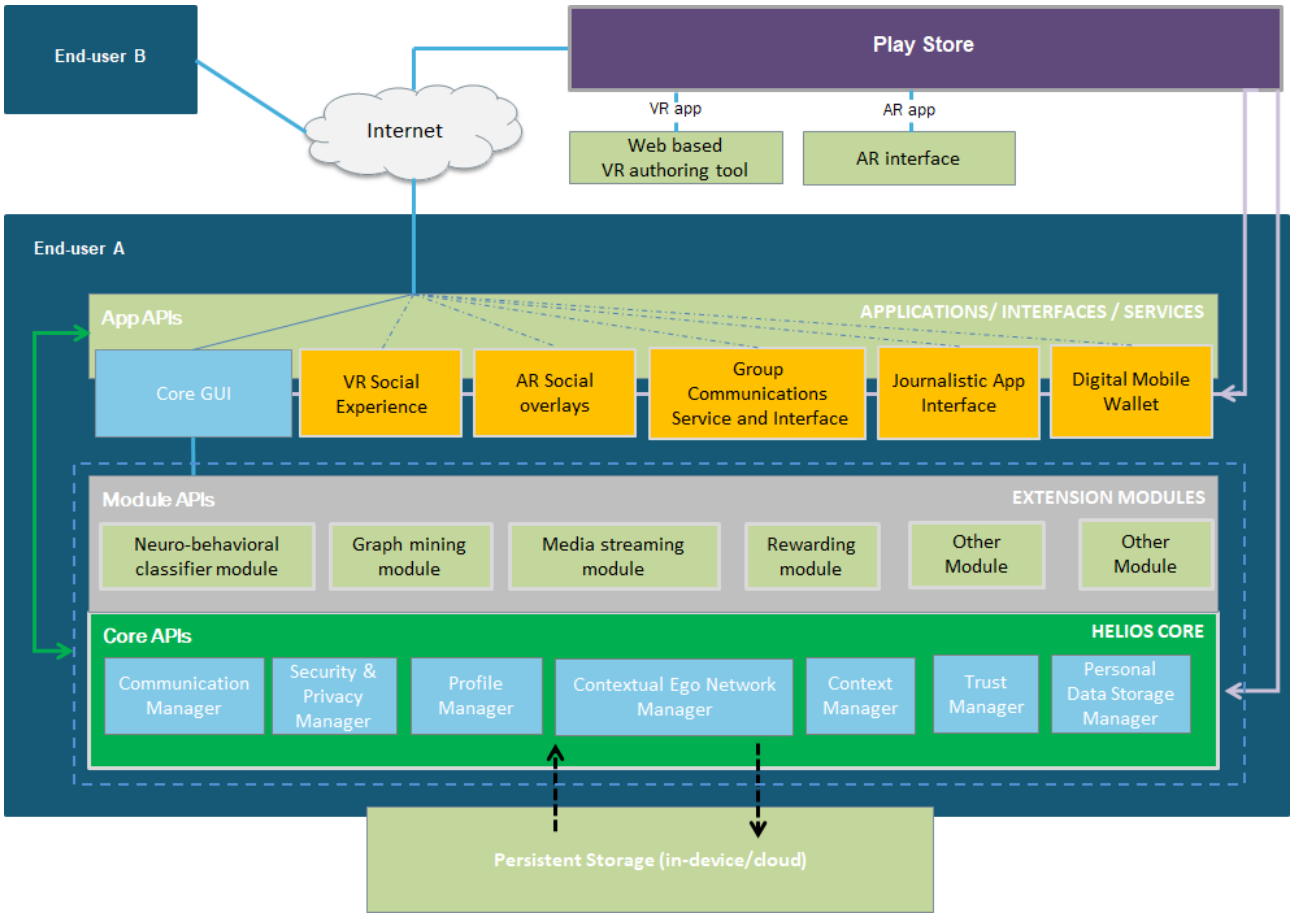

Figure 6. HELIOS core, extension modules and applications.

The following describes the general functionality of the core components depicted in **Figure 6**.

- **Core GUI:** Includes basic user interface for P2P messaging (like a generic messenger) / posting (similar to a wall) / reacting (e.g., like, love, share, repost, favourite, etc.). As such, it should be part of the HELIOS core application; it will also be described in the application section below.
- **Communication Manager:** Includes essential messaging functionalities, discovery of other peers, offering basic media functionalities, etc.
- **Security & Privacy Manager:** Includes, among others, access policies and privacy settings for user profile data. All the necessary functionality for managing user's security, including encryption functionalities, key management, etc.
- **Profile Manager:** Holds the user profile (schema). It takes explicit user input, can take device sensors input and can take input from modules and applications.
- **Contextual Ego Network Manager**: It is the structure that contains the local view of the Social Overlay of a node. This structure will be used by all the other modules to keep track of the changes in the Social Overlay and the social activity of the nodes in it.
- **Context Manager:** Handles all the context-related monitoring and reasoning for the user. Also includes information overload controls notifications management.
- **Trust Manager:** Takes care of calculating and updating trust scores per connection, per context, in time intervals.
- **Personal Data Storage Manager**: Takes care of the data access and layer abstraction for individual personal data storages the user might have, e.g., smart phone or cloud.

Extension Modules have APIs that can communicate with the core and with the apps. Application/services/interfaces have APIs that communicate with the core and/or one or more individual extension modules.

Figure 7 depicts possible communication paths between two applications (AR Social overlays and Journalistic App) can have with the HELIOS core (core components and the extension modules). Communication paths to/from other core components are implied, e.g., Trust Manager and others.

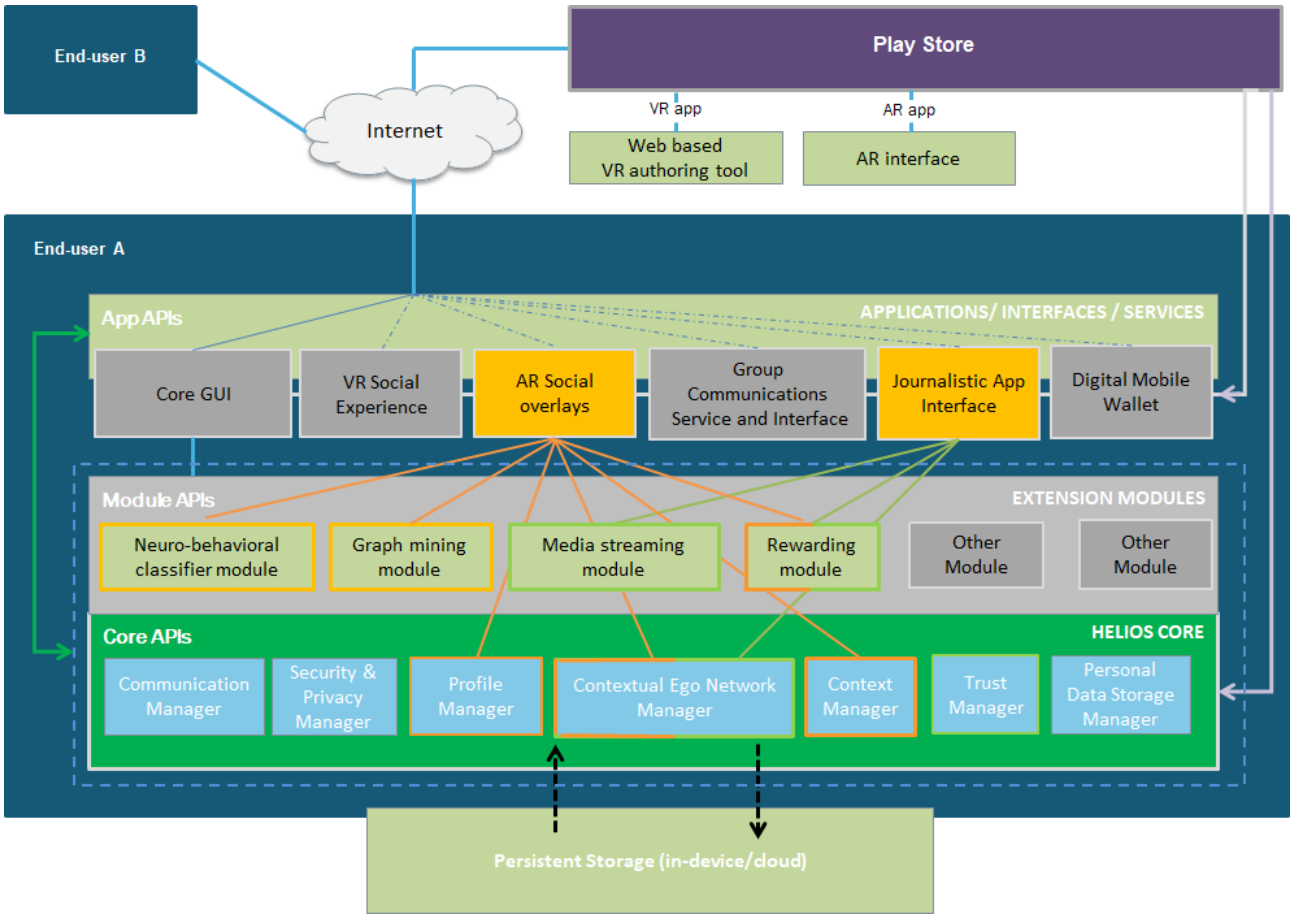

Figure 7. Example of two applications interacting with different core components and installed extension modules.

As can be seen in **Figure 7**, the extension modules and HELIOS core components can offer their functionality to multiple applications at a time. This is not considered as a performance bottleneck at this phase, as in the normal case the user will only be using one application at a time for the resource-intensive tasks, such as video streaming. The background applications can still be served for notifications, for instance.

4.3 HELIOS core components

A note on software reuse: While implementing the HELIOS core components, HELIOS will reuse existing open source protocols and components wherever possible. The consortium will decide if specific open source components will be integrated into the HELIOS system. In case that software component is maintained and updated by a third-party developer, considerations will be made on, e.g., how well it is updated, especially in terms of security. In addition, if modifications to the third-party component(s) are needed for the HELIOS functionalities, there is a risk of having to port the updates iteratively to HELIOS versions, unless the HELIOS modifications are accepted to the mainstream development of that specific component.

The following sections describe the high-level functionality of the currently defined HELIOS core components. Core components can evolve over time, in case some extension module is considered essential and is moved as a core component in the platform.

4.3.1 Communication Manager

The main component handling the communication in the HELIOS core is the Communication Manager. The applications or extension modules can request for discover, search, create, send, and receive messages in P2P networks using the API calls of it. Communication Manager uses the device's Wi-Fi, 3G/4G/5G or Bluetooth connectivity interfaces. The communication between peers might be, e.g., IP-based unicast, multicast, or broadcast transmission. The Communication Manager also handles the publish/subscribe type event messaging, if needed. The application or extension module can send/receive network traffic also by itself, if the P2P network is already created by the Communication Manager.

Communication Manager - P2P Messaging module

The messaging module providing access to the P2P network handles the lower level functionality to fulfil the requirements related to P2P features.

The module requires additional interface layer to the core - the interface between Communication Manager and the P2P Messaging module. This API handles transforming the upper level data from the core modules into the data that is supported by the P2P library, and vice versa.

4.3.2 Security & Privacy Manager

The **Security & Privacy Manager** includes the following functionalities/managers internally: *Key manager, Encryption manager, Access control by blockchain, Identity manager,* and *Personal data / Device access control manager.*

All the user's data in the personal data storage shall be encrypted. For efficiency, a symmetric encryption method like AES is preferred. The AES keys may be stored in the same data storage if they are encrypted using some other key.
The user may grant access to files to other users by revealing the key that was used to encrypt them and the location the files are stored in. The user may reveal the AES key by, e.g., storing the key to the data storage after encrypting it with the receiver’s public key. Alternatively, the user may send these encrypted files to another user and include, in the message, the used key encrypted by the intended receiver’s public encryption key. Only the receiver with the correct private key shall be able to retrieve the key needed to decrypt the content.

All messages and files stored in the data storage or sent to other users should be signed using the user's signing key. Anybody who has the user's public signature verification key may verify that the message is from the user and it has not been modified by somebody else. The public signature verification key should be given, together with the user's encryption key, to every other user they meet and want to communicate with.

In addition to protection of user data and communication, the security and privacy manager also protects data and communication of a HELIOS platform instance, e.g., two platform instances exchange data about the platform operations.

There should also be a distributed database of user public keys. From the database, users may retrieve keys of other users they want to communicate with.

Key manager
The key manager should store the keys that are generated by the encryption manager and will be needed later. It is possible that a different (AES) key is needed for almost every piece of content or at least for every context. In addition, the public keys of other users should be stored by the key manager. Some keys will be for symmetric cryptosystems (like AES), some for public key cryptosystems (like RSA).

Encryption manager
The encryption manager should generate cryptographic keys. These keys are needed to encrypt and decrypt the content user is generating, or signing content, or protecting integrity of the content by message authentication codes. These keys can be for example RSA, AES or HMAC keys.
It is also in charge of encrypting and decrypting messages/files using the selected encryption mechanisms. It should also be able to sign messages and verify electronic signatures made by other parties when their signature verification key is known. It should also be able to create message authentication codes that protect integrity of messages. In addition, encryption manager can support authentication of other entities.

Access control by blockchain
When the user content is very popular or the user wants to grant access to only certain other users (e.g., buying customers), blockchain can be used for access control. Blockchain makes it also easier to grant access only for a limited time. If some conditions are met, a user (or the content provider) can add a token (e.g., H(x), where H is a cryptographic hash function) to the blockchain. When the database server of the content provider sees this token in the blockchain, it grants the user (e.g., under the condition that the user provides the correct x) access to the content. The token may later be revoked. The details of this technology and methods for access control that use blockchain will be decided later, for instance, smart contracts are potentially used. What kind of blockchain technology to exploit, private or public, permissioned or permissionless, will also be decided later. Different approaches could be used according to different scenarios.

Access control is a critical feature of any computer system, especially a social network system such as HELIOS, as mentioned above. A blockchain implementation makes it easier for the rights to be granted only for a limited time, to be easily revoked later on (see [20]), but also makes it possible to know the users that have the right to access a given resource (provided we use a public blockchain, such as Alastria[21]).

HELIOS will investigate the blockchain technology most suitable for the different HELIOS scenarios and use cases. In the *HELIOS Online P2P*, where the user has connectivity to the Internet, a public blockchain such as Alastria could be exploited to store access control policies. In the *HELIOS Local P2P*, where there is no connectivity to the Internet, the previous possibility is no more feasible, and an

[20] "A blockchain based approach for the definition of auditable access control systems", Damiano Di Francesco Maesa, Paolo Mori, Laura Ricci, accepted for publication, Computers and Security, Elsevier, Computer & Security, Volume 84, July 2019, Pages 93--119.
[21] https://alastria.io/en/

alternative method should be envisioned. The peers in the local context can form a temporary blockchain among them, and this blockchain should be synchronised with the main blockchain when the node passes to the *HELIOS Online P2P* mode.

Identity manager

Uses key manager and encryption manager to create and verify persons and to associate keys for them. It is possible to learn other users' public keys by meeting them face-to-face, contacting them or from some distributed database of public keys.

Personal data / Device access control manager

This manager controls who can access the data in the user's data storage and what kind of access HELIOS has to the user's device. The user should be able, for example, to control how much the user's profile is visible to different groups of users. The user should also be able to decide how much access HELIOS needs; for example, the location, camera, microphone, movement sensors, other equipment and the files on the device. This manager can be used by Personal Data Storage Manager as well.

There are several methods for access control within HELIOS. All data should be locally encrypted to protect the data in memory. In addition, another layer of encryption can be used for access control in such a way that only the parties that are granted right to read the data have the keys needed to decrypt the data. Another method for access control is attribute-based access control (ABAC). Both methods have their pros and cons. The methods can also be combined. In addition, we may also utilise functionalities described in the access control by blockchain section above.

Other functionalities related to security and privacy

Other parts of the HELIOS platform that can incorporate security and privacy functions are the tamper resistant control data storage for maintaining integrity between two instances of the platform and the parts needed for peer anonymity. The tamper resistance could be achieved, for example, by trusted execution environment (TEE). Implementation of the security & privacy module can implement this in case needed.

Control data storage is needed to store data that the user may be tempted to alter if possible, for example, the number of reads of user’s posts or the number of “likes” or “thumbs ups”.

For peer anonymity, it may be required that the IP address of the user is only known to some set of peers, e.g., the friends.

For privacy preserving comparison of characteristics between two user data instances, secure multi-party computation methods can be provided. One option is to use tailor-made protocols directly between the two peers. Another option is to utilise a mutually trusted third party, which could be implemented by a trusted execution environment in one or both devices. The trusted party could also be an independent semi-honest entity. A semi-honest entity is a party that is assumed to faithfully follow the protocol but it is not forbidden to gather information while doing it. Thus, the protocol should be designed such that there is no useful information for the party to gather.

4.3.3 Profile Manager

The Profile Manager handles user account creation and creation of associated keys. At least one key pair is needed for signing and verification of electronic signatures and another public key-private key pair is needed for encryption and decryption. Other keys are needed for encrypting contents the user is producing and/or sending to other users. Uses Identity manager and personal data storage manager to handle profile information control, as well. This manager also interfaces with the functionality offered by, e.g., Security & Privacy Manager.

4.3.4 Contextual Ego Network Manager

The Contextual Ego Network is the structure used to model the social network of each user. It represents the people-centered view of the whole HELIOS Social Network. A Contextual Ego Network is a complex model organised in layers, where each layer represents a real user's context. Each layer, in turn, is implemented with an Ego Network model. The manager is the CEN structure proposed in T4.1 and T4.2. The CEN is a self-contained, fault tolerant structure. It is responsible for the setup of the data structure which models the multilayer nature of HELIOS, and the management of each singular layer. Layers can be populated by the means of adding/removing/updating alters and connections between the ego and its alters (friends), and the connections among alters. CEN layers are created and managed as per instructions offered by the Context Manager. The CEN also offers APIs to Interact with the layers, such as to retrieve information about the contexts and the alter nodes in each layer. The CEN is a passive structure which serves the requests to store in main memory the information about contexts and the alters as instructed by the other components of the platform. It can be instantiated as fault tolerant, which means that it keeps a registry of the recent modifications in order to restore them in case of an unexpected failure (such as when the device is forcefully turned off).

4.3.5 Context Manager

The Context manager handles all context-related monitoring and reasoning for the user. It provides a framework for context types and context detection with core implementations. Besides hosting the framework for extracting the current context, it will also need to have access to local (device-related) sensors and other context sources, since those are building the context extraction properties. The Context manager can access sensors with the help of a Sensor Manager, the core of which is built inside the context manager. Helios extension modules can provide extensions to the core functionality, and sensor and context core vocabulary.

The Notification system/manager (and information overload control), as part of the Context Manager, internally limits the amount of incoming message alerts for the user. Limiting of the alerts depends on the state of the user's momentary ability and willingness to process incoming messages, eventually related to avoiding information overload. Users are encouraged to read primarily those messages that are relevant at a particular time/ in a particular context. Since there is a contextual social network active at the time of reading a message, the information about reading a message is propagated to alters in the context. For example, a short reaction time of a particular message is a sign of importance that will be propagated to alters, if they have received the same message.

4.3.6 Trust Manager

The Trust Manager handles the trust model and its management. The model takes into account the information concerning the behaviour of the users expressed with the interactions, and other important

features which will be identified in the task T4.5. Two important features are represented by the output of the task T4.6 and the proximity information that could be retrieved by exploiting the functionalities of the HELIOS core modules. The model will be able to initialise the trust value and manage the evolution over time, according to recent activities and their importance. The trust score computed by the trust model is a private information stored in the CEN as a parameter of the connection between the ego and an alter. Since trust is related to the context, an alter will have different trust scores, one for each context in which it is included.

Considerations for trustworthiness and confidence. Two important metrics in terms of the trust score model utilised to represent trust are trustworthiness and confidence. The former is composed of an aggregation of the multiple evaluations towards the subjects regarding specific applications with each measurement being a single piece of data, contributing to the evaluation of trustworthiness. Confidence, on the other hand, is the level of certainty in respect to the trustworthiness evaluation, since confidence varies depending on the number and type of measurements used. In terms of the computational problem of trust, usually, those levels are computed with either discrete or continuous values according to the underlying algorithm. Threshold values can be accommodated to measure trust as well in order to establish the level (amount) of information to be shared from each user in their interaction. To measure trust and the degree of trust between users, a broad classification into global and local metrics will be utilised.

Trust is closely related to the strength of the connection in the Contextual Ego Network, with trust values being highly context-dependent, indeed a change of context usually requires a change of trust values as well. As such, the trust management is interdependent on the Contextual Ego Network. The Trust Manager computes and updates the trust scores of the ego at certain time intervals (keeping in mind the energy requirements of each device) as it will be defined in T4.5 and provides the necessary values in order to establish the required data sharing levels (amount of data in terms of privacy) shared with other nodes. For nodes without established trust value, an initial trust value between two nodes in a particular context is provided. This helps to remove the cold start problem The Trust Manager will rely on several algorithms to compute trust values using global (when and if available) and local features. To this end, the Trust Manager gets data from other modules through their APIs. It will furthermore offer APIs to communicate with the CEN library.

4.3.7 Personal Data Storage Manager

As the personal data storage is an important concept in terms of the HELIOS platform, this section describes the general concept of the Personal Data Storage Manager. Personal data will be accessed through a unified API that is agnostic to the storage type (local, cloud, etc.).

By default, the user will have their personal data storage on their personal mobile phone, and the phone is connected to the Internet 24/7. That is, the mobile phone is storing all the user's data and the device is persistently in the HELIOS network. The profile information, context information, social graph information, etc. In addition, it will store the actual data produced while interacting with other users in the network, i.e., messages, media content, etc. The user has full access over their data and by the privacy settings makes it available to other users in the HELIOS platform.

However, not all users are not willing to be connected or they have such content that is not suitable for a mobile storage. In case the user wishes to store the data in a cloud (be that personal or public), the HELIOS platform supports this by abstracting the personal data storage interface for the applications

and the user. The user only has to specify what personal data storage is being used and the platform takes this data storage into use. The supported cloud storages will be decided at a later phase.

The user may wish to select if personal or profile data is stored in their mobile phone and other content related to messaging or other interactions to be stored in a cloud system (in order to save space in the mobile device). This should be supported by the HELIOS platform.

The data owner in the different communications/context settings is the person or device creating that context/chat in the first place. The data provided by other users will be stored in that personal data storage. The data provided by other users can directly be saved in that personal data storage or the user might want to only provide a link to a resource available in the users own personal data storage. In this case, everyone accessing the content in the first context must access the linked data from the specified personal data storage - and thus have access to it.

Each user may have several personal data storages for different purposes, to meet different requirements related to, for instance:

- Privacy
- Availability
- Reliability
- Storage capacity
- Bandwidth
- Backup strategies

Data-abstraction layer should support the following concepts:

- Local data lake
- Cloud data lake
- Access control mechanisms (local) - Outside world access control is handled via HELIOS core.

4.4 HELIOS extension modules

The HELIOS extension modules can be developed by third-party developers to offer additional features on top of HELIOS core components. The HELIOS project/community can also provide more privileged extension modules that are signed with the same key as the HELIOS core in order to provide the user trusted extension modules. The extension modules are installed from the Google Play store. The following section describes foreseen extension modules to be implemented during the project.

Neuro-behavioural classifier module

A set of algorithms are implemented as an extension module to analyse the emotional statements of the users with two main aims. Firstly, it determines an emotional score between an ego and an alter, giving this info to feed the trust model (e.g., if one user harasses another, the neuro-behavioural

classifier detects it, which is then evaluated and reflected by the probable decrease of the trust score in the computed time interval). Secondly, it analyses the content shared creating metrics to give feedback on how the users response to a specific stimulus (e.g. analysing the reactions of the users to an AR content using facial recognition, showing this indicator attached to the content). The module includes sentiment analysis, voice analysis, image recognition and facial recognition. The module is validated in a lab environment with real users using neuro-physiological signals.

Graph mining module

A bundle of algorithms that are implemented and deployed as an extension module, allowing graph mining techniques to be applied on the user's local social graph. The module takes as input information stored in the CEN of the ego, such as social relationships and interactions, and outputs several higher-level node and ego network properties as a result of analysis taking place over the user's network. The outcomes can be forwarded to the core, in order to populate the user profile with additional information that is not explicitly provided by the user (e.g., interests, communities, etc.), or can be used by an upper layer application for building better, more intelligent services (e.g., group communication suggestions, link predictions, etc.).

Media streaming module

This extension module handles the P2P streaming and media distribution functionalities.

This HELIOS extension module for media streaming and distributions is designed to use its features individually having all of them separated in different libraries improving the modularity of the overall system.

The module itself requires external (to the mobile device) resources (i.e., to host a Real-Time Messaging Protocol (RTMP) streaming engine, a WebRTC signaling or a TUS service) to execute some of the tasks that are not possible to accomplish in a mobile device like the video transcoding or the HTTP server. This device is called Personal Storage or Personal Cloud and it is packed with all the necessary software to be provided to the users. In depth:

- A Live adaptive streaming service allows the user to create a live video to be adaptively broadcasted to certain contacts present in the HELIOS context. The video from the device's camera is sent via RTMP[22] to the Personal Storage and there it is transcoded and adapted to HLS[23] (HTTP Live Streaming) with different bitrates to be consumed by the rest of the users. This functionality requires on-the-fly encoding at multiple bitrates to ensure a high Quality of Experience (QoE).
- A Video Call service that provides video call functionality among two users that belong to the same context. This video call is done over to the WebRTC[24] protocol which is an open-source project to provide the web browsers and mobile phones real-time communication via APIs. The following components must be deployed on the Personal Cloud:
    - Signalling service. This module is in charge of starting the communication between peers. HELIOS must identify the realisation of this functionality, considering the decentralised project approach. The functionality of signalling is always required. But can be done in a

[22] https://www.adobe.com/devnet/rtmp.html
[23] https://developer.apple.com/streaming/
[24] https://webrtc.org/

distributed way without a central server in the case the ad-hoc network is not connected to the Internet.
    - Session Traversal Utilities for NAT (STUN) service. This service includes methods to enable the communication even in the presence of network address translators. This functionality is not needed when the peers are in the same network.
    - Traversal Using Relays around NAT (TURN) service. In some cases, both peers are behind NATs and firewalls that prevent the direct connection. A TURN service should be used. The use of a TURN service is transparent to the user, it is decided by the signalling service. As all the contents shall pass through the TURN service, that acts as a relay, using a TURN the transmission is not fully P2P, but it is the only workable solution sometimes. As this case will always be connected to the Internet, the TURN server can be installed in the Personal Storage of one of the peers.
- The Video on Demand distribution (VoD) service, broadcasts video contents previously acquired. This streaming modality can be adaptive if the live streaming is adaptive streaming. This requires a previous step of encoding at multiple rates.
- The file transfer functionality is provided to share multimedia content such as images (photos), audio (songs) or videos (even 360) with other users. The content is uploaded to the Personal Storage using a tus[25] container, this component enables resumable uploads in case of lost connection. Once the file is uploaded, it can be downloaded via HTTP.
- A video player is also available inside the Media Streaming Extension Module. This video player can play HLS streams or local files present in the mobile phones. The player can be customised for different use cases or situations.
- Premium Live and VoD streaming distribution and support for multiple connected users without decreasing users Quality of Experience (QoE). The distribution of high quality content can generate revenue. For this reason, the extension set will include billing procedures and modules, as explained below. To implement this functionality HELIOS will integrate a reliable software to set up a CDN architecture.

HELIOS will provide a rewarding system between peers to enable monetisation options for premium content. This rewarding system will verify the same HELIOS principles, avoiding information centralisation.

- Video on demand distribution requires a media storage service and an API to access the content. This functionality can also be used for the image storage and exchange. Beyond storage capabilities, this module could also integrate processing (encoding) capabilities to produce the multiple qualities required by adaptive streaming.

Rewarding module

During the last years, the amount of content creation through social media has grown rapidly. Every single day, we consume and create content ranging from articles to videos, images, etc. from a wide number of social media platforms such as Facebook, Twitter, YouTube and so on. However, most of these social networks fail in providing a fair and transparent rewarding system. HELIOS rewarding model (see D4.4 “Define rewarding methodologies”) will leverage on Helios token as a fundamental unit of

[25] https://tus.io/

account for rewards in the HELIOS network. HELIOS users will collect Helios tokens eligible for reward-valuable actions in the network and contributions to the content creation.

Daily computation of the total number of tokens assigned to users is performed off chain but REST API is connected to the blockchain and all rewards are initially transacted via the Quorum Network through smart contracts.

The main function of the decentralized reward system is to incentivise Helios users in the use of the platform. Users will be able to collect this incentive, in the form of a fungible token, and will be able to spend it for benefits within the platform (access to premium features, access to premium content, access to external services).

Tokens are often used as a representation of an asset or utility and can be used to reward HELIOS users for several reasons. For instance, tokens can incentivise the sharing of information useful in a particular context, by providing the user who has published the information with a set of tokens. These tokens could later be used, e.g., for obtaining other HELIOS services, for instance obtaining the access to information that other users only wish to share with some users.

Sensor manager extension

This module is considered as one of the Other extension modules. It provides an extension to the core Sensor Manager (within the Context Manager) and would make outside sensors available via Bluetooth/BLE. This module would allow abstracting the connection to outside sensors, receiving data and providing it for the HELIOS core and/or applications.

4.5 HELIOS applications

The applications developed for the HELIOS platform use the functionalities and structured information provided by the HELIOS core and installed extension modules to communicate with other HELIOS applications (i.e., other HELIOS nodes/people). The applications can be developed by anyone in the community (i.e., third party developer), who is looking to provide novel user-centric services for the HELIOS platform. The platform's modularity allows for indefinite development of social network services at the application layer.

The following social network services are planned to be developed during the project's lifecycle (some of them are the actual end-user applications running in the users' Android mobile phone, while others are used to develop the services for the applications):

- **Core GUI:** Use HELIOS core (and possibly extension modules) APIs to send and receive messages. Associate user profile via HELIOS core. Provide the minimalistic GUI for the user to communicate with other users and define the user profile. It should be provided for the users when they install the HELIOS core package, as a minimal user interface.
- **Group communications services:** A customisable service that caters for different group communications needs, leveraging information about active nodes and contexts in a network, relationships and associations between actors. When applicable, user profile information is utilised, deriving from the HELIOS core and from technologies built as modules on top of the core. Processing of all information is envisioned to take place on each user's personal data

storage. In cases where technical limitations hinder the processing, alternative distributed processing schemes will be explored.

- **VR authoring tool:** A suitable for the project's concept, IDE that is portable and light-weighted based on Web 3D frameworks, allowing easy authoring of VR experiences. The authoring process will be facilitated by a web interface that enables building a 3D space out of a template, positioning of 3D items in a space and assigning behaviours onto them as a category with inherited properties. This will allow hiding any programming details that are not suitable for non-experts trying to design a VR experience. More specifically, the front-end interfaces of the authoring tool will be developed using a WebGL library such as open source Three.js (http://threejs.org) that allows the construction widgets and visualization of 3D information. The backend will be developed in a game engine such as Unity3D that allows compiling VR experiences in high-quality runtime format, including also Android Package (APK) format. Through the authoring tool, curators and users can create virtual experiences that allow for multiple user connection and interaction in virtual spaces. To achieve this result, an executable (VR app) is created through the authoring tool that enables this behaviour and acts as the main component for the overall shared virtual experience. The necessary information for interlinking HELIOS core with the VR generated application will be inserted during authoring time.
- **VR app:** An executable (i.e., compiled VR experiences) that was built using the web-based VR authoring tool. A VR application can be accessed to be installed via a URL link (deeplink) either from Google Play Store or from another web repository. The apps can be configured to allow multiple players/users to participate in real-time and interact through text or voice in a VR environment. In order to ensure a frictionless, robust, multiplayer VR experience of high quality, the network architecture followed is aligned with current industry standards and strategies of client-server protocol where all clients exchange information through a lightweight server (advanced relay), where GDPR compliant communications are established for multiplayer action synchronisation/orchestration purposes. Profile data and social network information available through the core APIs are to be leveraged to, e.g. identify and match 3D avatars of users in the VR space to known connections from the ego-network. One important aspect of virtual experiences is the ability to keep and analyse information concerning the experience of the user. User activity while inside the VR experience (e.g., movement, interactions, as well as metrics, such as the duration spent, and the options taken) can be transmitted back to the HELIOS core to populate or refine the user's profile and/or update trust with known or new connections.
- **AR overlays:** An interface capable of sharing and overlaying content, information and experiences over objects in a physical space. A combination of auditory and visual cues can be presented to the user, combining 2D and 3D content in a way that does not limit the situational awareness or in other ways hinder the current actions off the user. Off-the-shelf hardware and open source libraries and tools will be used when possible. AR overlays communicates with neuro-behavioural classifier module to characterise the reaction of how people are reacting with the AR content, showing it in the interface.
- **Journalistic App:** The journalistic mobile application integrates STXT's broadcast grade workflow engine, which can perform the Orchestration of Microservices. A specific workflow will be designed via a visual interface using Business Process Management Notation (BPMN). The transactional information can be consolidated for billing. An Ethereum Smart Contract is also to be implemented. Content rewarding (e.g., for premium content in the context of citizen journalism, or content that is made available by HELIOS users to be utilised by publishing/news

companies) may be enabled through the respective extension module related to payment technologies, based on e.g., blockchain.

- **Digital Mobile Wallet:** The Digital Wallet allows end-users to access or observe tokens and services related to their rewardable activities in HELIOS Apps. They identify their required Decentralized application (DApp) and create or import an identity. Next, they acquire tokens in order to buy, transfer and use services (e.g., streaming TV). The Digital Mobile Wallet Application covers the following core functions for users:
    - Account and identity registration
    - Receipt of HELIOS Tokens
    - Send HELIOS Tokens to other users
- **Other applications possibly required for piloting**: These applications will be defined in more detail later in the project, when implementing the selected use cases for piloting. The applications and services described above already fulfil a large set of features required to demonstrate the functionality of the HELIOS platform and ecosystem.

5. HELIOS reference implementation: Android considerations

HELIOS project provides a reference implementation for the Android OS. The following sections present some Android features and restrictions that must be taken into account, while designing and implementing the HELIOS platform architecture for the Android OS. It also presents the P2P library options and considerations for the platform.

5.1 Android connectivity

For implementing the HELIOS Local P2P mode, Android supports Wi-Fi Direct and Wi-Fi Aware (Android 8.0/API level 26 and greater). Wi-Fi Aware is also known as Neighbor Awareness Networking (NAN) [26].

Android also supports Bluetooth and Bluetooth Low Energy (BLE), which can be used to exchange messages/content of small size.

5.2 Android IPC/RPC

When thinking about interactions between different HELIOS core modules and/or HELIOS core and HELIOS applications inside the user's phone (i.e., not between HELIOS users), there are a few options from the Android side.

[26] https://developer.android.com/guide/topics/connectivity/wifi-aware

Android offers multiple options for inter-process communications[27] (IPC) or RPC communications. Instead of using traditional techniques, such as network sockets (that are discouraged to be used), these options can be used in the Android implementation.

- One must consider how easily these can be implemented/ported in/to other environments. Although, if JSON messaging is used in the IPC (mostly) with named API functions, that functionality should be quite easy to port into a REST (Representational state transfer) implementation, for instance.

- Another option is that HELIOS uses network sockets for the local communications, but then HELIOS has to take care of the security and privacy even more carefully. Naturally, when communicating with other HELIOS nodes, the same kind of security techniques will have to be used.

When using the recommended binder IPC, one has to take into account the transaction size limit, which limits using it for sending large amounts of data blobs at once.

- **Binder Transaction size**: "The Binder transaction buffer has a limited fixed size, currently 1Mb, which is shared by all transactions in progress for the process. Consequently, this exception can be thrown when there are many transactions in progress even when most of the individual transactions are of moderate size."[28]

For sending/receiving large files (images, videos, etc.) HELIOS could use FileProvider[29] with temporary permissions.

5.3 P2P messaging library

Two P2P messaging library options from the previously listed options in Section 2.2 are discussed in the following (in terms of their pros and cons).

BRAMBLE is currently used by the early release of the "helios. TALK" application (one of the use cases of the HELIOS project) as a placeholder on peer-to-peer communication in the early stages of the development in the project due to a number of advantages. BRAMBLE is almost a 10 years old project, Android first, written in Java, and it supports delay-tolerance by design, privacy-by-design, security-by-design. It also offers a protocol for data synchronisation and is released under GNU General Public Licence[30]. Nevertheless, BRAMBLE also suffers from some important drawbacks; it lacks good documentation and most importantly an effective peer discovery mechanism. As a consortium in the early stages of the project, we decided to use native Java for the development of the core, extension modules and Android applications. The availability of BRAMBLE in Java and the fact that it is designed as an Android first stack of protocols were the main reasons to choose it as a placeholder.

libp2p is currently used by the HELIOS core Communication Manager and the internal P2P Messaging module. It has multiple implementation languages, from which JS and Go versions are the most

[27] https://developer.android.com/training/articles/security-tips.html#IPC
[28] https://developer.android.com/reference/android/os/TransactionTooLargeException
[29] https://developer.android.com/reference/androidx/core/content/FileProvider.html
[30] https://www.gnu.org/licenses/gpl-3.0.en.html

mature ones. Current implementation of the messaging module is with JS, but the Go version development has also started, because the JS version lacks behind of the documentation and some of the modules are not as mature as the Go counterparts, as per our investigation.

The drawback with libp2p in Android is that the Java version is not mature at all, and in order to use libp2p in Android, one has to use one of the other language versions. In general, documentation is a bit lagging behind the implementations. JS version with all the dependencies in Android makes the messaging module very large in terms of size (in addition there might be problems with license clashing). JS version is also not as mature as the Go version, specifically there is wildly varying level of maturity between components.

Annex 1: HELIOS Objectives

The following table is a direct excerpt from the original HELIOS Objectives.

Objective ID	Objective description
OBJ1	Create a Peer-to-Peer communications network for mobile devices, which can deliver messages over legacy IP network. *[Measurable outcome: In a setup of three Android phones, without any centralized server, each phone attached to a different point-of-presence, each phone can send a message simultaneously to two other phones]*
OBJ2	Create a Peer-to-Peer communications network for mobile devices, which can operate over legacy IP network in such a way that if one of the phone is disconnected from the network, it will receive the message when connecting again. *[Measurable outcome: In a setup of OBJ1, one of the receiving phones is switched to flight mode, when a message is sent from another phone, and that phone receives the message after the flight mode is switched off, without any other interventions.]*
OBJ3	Validate the platform with real users that sharing common interests in communicating with each other. *[Measurable outcome: The setup of OBJ2 will be used by a group of at least 20 people during a period of at least one month, for instance by VTT's Summer Trainees (see VTT Support Letter form]*
OBJ4	HELIOS will be based on Open Source principles. *[Measurable outcome: All source codes for any HELIOS software component (core, modules, apps) will be publicly available over Internet from at least one server.]*
OBJ5	HELIOS platform can deliver payments. *[Measurable outcome: In the setup of OBJ2, having different users for each mobile phone, one of the users can send one unit of real or virtual currency to another user.]*
OBJ6	Functionality of HELIOS platform can be expanded by adding new modules into it. *[Measurable outcome: In the setup of OBJ2, all phones are using identical HELIOS communication cores, but only two of them have an extra module that enables sharing 360 degree vision in both directions.]*
OBJ7	HELIOS platform supports content co-creation between prosumers and professional journalists. *[Measurable outcome: One HELIOS platform user has a module that receives a message from a newsroom production system, the user can modify the message and send it back to the newsroom system.]*
OBJ8	HELIOS supports analytical data over content usage. *[Measurable outcome: In the setup of OBJ2, One HELIOS platform user sends a message to both other users. Again, one mobile phone is in flight mode. The user who has sent the message is able to see that the message has reached one other user. When the flight mode has been switched off, he/she is able to see that the message has reached two other users.]*
OBJ9	HELIOS supports correct analytical data over content usage. *[Measurable outcome: In the setup of OBJ8, the platform maintains the information of reader count in a cryptographically secure repository into which the user has no access, and when the information is provided from the repository, it is presented with attestation.]*
OBJ10	HELIOS supports distribution of encrypted content and has a selective key distribution. This key distribution may be based on blockchain technologies. *[Measurable outcome: In the setup of OBJ2, one user sends the same message to both recipients in encrypted format and sends keys to only one of the recipients. The recipient*

	who gets the keys receives the message without extra interventions, while the other user is not able to see the contents of the message.]
OBJ11	HELIOS consortium is utilizing creative power of artist to develop new concepts and user interfaces. *[Measurable outcome: The Design Fiction process will produce at least one unforeseen social media application and at least one novel user interface concept that at least 50% of focus groups finds interesting.]*
OBJ12	In HELIOS platform there will be at least one module that enables file-based near-real-time video sharing. *[Measurable outcome: In the setup of OBJ2, one user receives a video clip that originates from both other mobile phones in a way that neither of the other phones have to provide more than 80% of the video clip.]*
OBJ13	HELIOS platform provides real-time geolocation data between peers. *[Measurable outcome: In the setup of OBJ6, one of the users of sending 360 vision is moving and the movement data is transferred to the recipient who gets an additional map image.]*
OBJ14	HELIOS platform is using context-based adaptivity to control frequency of given message notifications. *[Measurable outcome: In the setup of OBJ1, one of the users of sends two messages to both of the other users in 1 minute intervals, one other user being in the same context with the sender receives notifications of both messages immediately, while the third being in different context receives one notification that contains both of the messages.]*
OBJ15	HELIOS project creates such business models for prosumers, including crowd-sourcing models that can be implemented in HELIOS platform. *[Measurable outcome: In HELIOS project at least 2 such business models are created to support prosumers, at least one involving crowd-sourcing model that can be utilized in HELSIO platform using software components that are developed in the project.]*

Annex 2: HELIOS Core APIs

This annex describes the current versions of the main APIs of the HELIOS platform. The main purpose of this annex is to identify the public APIs available to the extension modules and the applications to access the core functionality in order to provide the social services to the users. The current version also includes some APIs that might be changed to private APIs between core components.

2.1 Key interfaces

This section describes the key interfaces between the core, extension modules and the applications. The detailed access control/authorisation of these interfaces will be made during the agile development process. The HELIOS project foresees at least the following interface enforcements in the Android implementation:

- Extension module <--> Core: Permission enforcement on separate interfaces (optional).
- Application <--> Core: Permission enforcement on separate interfaces.
- Application <--> Extension module (optional): Permission enforcement on separate interfaces.
- Core <--> Personal Data Storage: Permission enforcement and access control/authorisation on cloud storage.

The main interface between Application and Core can be, e.g., AIDL IPC-connection enforced by the techniques explained earlier in the document. The main communication would be serialised request-reply or command-reply messages sent between the Application and the Core to allow extensibility in terms of applications. For example, new application can then easily define a new protocol (see Section 4.1.3) for the underlying P2P module.

2.2 Ego management API

Ego Management API provides access to various HELIOS settings that HELIOS users can use to configure their environment. The API should have operations to configure the following properties.

- User profile management and settings
- User profile privacy settings

2.3 Contextual Ego Network API

The Contextual Ego Network API provides access to read and edit the information in the Contextual Ego Network. The full list of the API provided by this module are presented in D4.3.

2.4 Context management API

The Context Management API provides methods for accessing and managing contexts. Further, a context can be associated with a layer of CEN using the Contextual Ego Network API. The core methods include creating and modifying contexts and context attributes, checking the state of context, and obtaining notifications about changes in context values, such as:

- CreateContext - create a new context
- DeleteContext - delete context
- IsActive - check if the context is active
- AddAttribute - add a context attribute, which maybe bound into several context sources
- RemoveAttribute - remove context attribute
- RegisterContextListener - register to get notifications about context active value changes
- UnregisterContextListener - unregister notifications
- GetContextListeners - get registered listeners of the context

The core SensorManager API methods:

- RegisterValueListener, UnregisterValueListener - register/unregister sensor value update notifications
- StartUpdates, StopUpdates - start/stop updates from the sensor

2.5 Trust Manager API

The Trust Manager API provides methods for computing the trust score between an alter and ego in a context having the following basic functionality such as:

- setTrust - Initialise the initial trust value between two nodes in a particular context.
- getTrustValue - return a reference to the current trust value such that it can be modified directly and the timestamp of last value update
- updateTrust - Update the trust value between two nodes in the specific context.
- computeTrust - Compute the new trust value

The entire APIs and the specific algorithms for trust calculation will be defined within the project deliverable D4.5 “Computational Trust”.

2.6 Messaging API

Messaging API is used to communicate with other HELIOS nodes (users). The following basic low-level operations could be supported. Addressing and encryption issues are taken into account as well, which will be part of the Security & Privacy APIs.

- Connect - create a connection. This could be either point-to-point connection or connection to a group.
- Send - send a message. This could be either one-to-one or one-to-many message.
- Receive - receive a message (e.g., direct connection between peers).
- Resolve - resolve peer info (e.g., HELIOS node ID to a P2P network ID).
- Search - search/discover connections based on attributes e.g., location, interest, name, etc. Search can be accomplished depending on the P2P library, e.g., by Distributed Hash Table (DHT) functionalities.
- Publish - send a message to a channel (optional) .
- Subscribe - subscribe to a message channel (optional) .
- Distributed Hash Table (DHT) functionalities - provide services based on DHT (optional).

.

2.7 Personal Data Storage API

HELIOS personal data storage API is used to load and store data from persistent storage. The API provides a storage abstraction for HELIOS applications without binding these API calls to specific storage location. HELIOS users are able to configure different persistent storage options like personal cloud storage or utilising storage of the local device. Distinguishing several kinds of content (pictures, videos, audio clips, etc.).

Public methods related to media/file content could be supported.

- ContentUpload
- ContentDownload
- ContentDelete
- ContentEncoding
- ContentSearch / ContentList

The storage API could implement lower file abstraction operations for internal modules (optional):

- open - open a file for read/write/append. If the file does not exist it will be created. The call should return a file handle that is used in other file operations
- read - read a chunk of data from a file
- write - write a chunk of data to a file
- close - close a file handle
- seek - reposition the open file handle
- stat - return information about a file
- No-sql like operations: write, select (execute), etc. for time series visualisation of context/graph

Additionally, the API could also provide directory level operations (if directory abstraction will be supported, optional):

- mkdir - make new directory
- mv - rename a file or a directory
- rmdir - delete a directory
- rm - delete a file

The API could also provide web server like operations for files (optional)

- get - read a file as a whole
- put - store a file as a whole

There may also be other needs e.g., providing keys to encrypted storage - these will be handled with the security and privacy API methods. Pathnames for files in persistent storage should be defined as well.

.

2.8 Security & Privacy APIs

Security & privacy API could expose parts of these functionalities to the applications. Access to these APIs depends on the interfaces between the applications and the core, extension modules.

Encryption manager API

- Encrypt
- Decrypt

- Sign
- VerifySignature
- CreateMAC
- VerifyMAC
- CreateAuthenticationChallenge
- CreateResponseToAuthenticationChallenge
- VerifyResponseToAuthenticationChallenge

Key manager API

- StoreKey
- RetrieveKey

Access control API

- ModifyDeviceAccess
- CheckDeviceAccess
- SetProfileVisibility
- CheckProfileVisibility
- SetPostVisibility
- CheckPostVisibility
- SetAccessRules
- GetAccessRules
- RequestAccess

2.9 Management API (optional)

The management API could be used to control and authorise different extension modules and applications to communicate with the core. It could also be used to authorize individual APIs of the core and or control the access to personally identifiable information (PII). In the Android implementation this API might not be needed, since permissions can be enforced for individual interfaces, thus providing a high-level management API for the modules and users to configure the access control and authorisation.

- Register - register a module
- Unregister - unregister a module
- Search - search module whether it exists
- Authorise - authorise access to certain features of the core

.