D3.6 Third Parties enabling APIs v1

WP3 – Cybersecurity risk assessment and Beyond – Sphinx Intelligence

Version: 1.00

SPHINX

A Universal Cyber Security Toolkit for Health-Care Industry

Disclaimer

Copyright message

Document information

Grant Agreement Number	826183	Acronym	SPHINX		
Full Title	A Universal Cyber Security Toolkit for Health-Care Industry				
Topic	SU-TDS-02-2018 Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures				
Funding scheme	RIA - Research and Innovation action				
Start Date	1st January 2019	**Duration**	36 months		
Project URL	http://sphinx-project.eu/				
EU Project Officer	Reza RAZAVI (CNECT/H/03)				
Project Coordinator	Dimitris Askounis, National Technical University of Athens - NTUA				
Deliverable	D3.6 – Third Parties enabling APIs v1				
Work Package	WP3 – Cybersecurity risk assessment and Beyond – Sphinx Intelligence				
Date of Delivery	**Contractual**	M18	**Actual**	M18	
Nature	R - Report	**Dissemination Level**	P - Public		
Lead Beneficiary	EDGE				
Responsible Author	Marco Manso	**Email**	marco@edgeneering.eu		
		Phone	-		
Reviewer(s):	FINT; PDMFC				
Keywords	API; Third-parties equipment and services; SPHINX sandboxed environment; Secure and trusted plug and play framework				

Document History

Version	Issue Date	Stage	Changes	Contributor
0.1	18-10-2019	Draft	Table of Contents	Marco Manso (EDGE)
0.2	04-11-2019	Draft	Content creation	SPHINX Partners
0.3	25-04-2020	Draft	Internal Review 1	Marco Manso (EDGE), José Pires (EDGE)
0.4	01-06-2020	Draft	Content consolidation	Ilia Pietri (ICOM), Yannis Nikoloudakis (HMU), Stylianos Karagiannis (PDMFC)
0.5	17-06-2020	Draft	Internal Review 2	Marco Manso(EDGE), José Pires (EDGE)
0.6	22-06-2020	Draft	Review 1	Dimitris Apostolakis (FINT)
0.7	23-06-2020	Draft	Review 2	Luís Ribeiro (PDMFC)
0.8	24-06-2020	Pre-final	Update to reflect the reviewers' comments	Marco Manso and Bárbara Guerra (EDGE)
0.9	29-06-2020	Pre-final	Quality Control	George Doukas (NTUA), Michael Kontoulis (NTUA)
1.0	29-06-2020	Final	Final	Christos Ntanos (NTUA)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Executive Summary

The SPHINX Application Programming Interface (API) for Third Parties (S-API) enables third-party solution providers to access and interact with the SPHINX Platform and its components. Subject to authentication, authorisation and using end-to-end encryption, S-API exposes advanced cybersecurity functionalities implemented by the SPHINX components, such as device/application certification, threat registry notification and anomaly detection. S-API therefore brings to SPHINX an easy integration with external components and the possibility for third-parties to extend existing SPHINX functionalities and incorporate additional functions. S-API also brings additional exploitation opportunities related with third-party's services.

This document presents the detailed design for the SPHINX S-API component, following the component's introduction in the SPHINX architecture deliverable (D2.6 - SPHINX Architecture v2) [1]. It extends the technical specifications defined in [1] with the specific requirements for the component's development, addressing its specificities, as well as interface specifications between S-API and the SPHINX components and third-parties. Importantly, it describes S-API interactions with third-parties and SPHINX services, which serve as a basis for the technical implementation of the S-API component in SPHINX.

The next iteration of this deliverable (D3.12) will incorporate refinements and updates of the S-API component, resulting from the implementation work to be performed throughout the duration of the task.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Abbreviations

AD – Anomaly Detection

AM – Administration and Management

AP – Anonymisation and Privacy

API – Application Programming Interface

BBTR – Blockchain-based Threat Registry

DD – Detailed Design

DTM – Data Traffic Monitoring

FDCE – Forensic Data Collection Engine

GDPR – General Data Protection Regulation

HE – Homomorphic Encryption

HTTP – HyperText Transfer Protocol

ID – IDentification

IP – Internet Protocol

IT – Information Technology

JSON – JavaScript Object Notation

REST – REpresentational State Transfer

S-API – SPHINX API for Third Parties

SB – Sandbox

SIEM – Security Information and Event Management

SM – Service Manager

TPCF – Third-Party SPHINX Certification Functions

TPMF – Third-Party Management Functions

TPSAF – Third-Party Service Access Functions

TTL – Time-To-Leave

URL – Uniform Resource Locator

WP – Work Package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

1 Introduction

1.1 Purpose and scope

This document, named *D3.6 - Third Parties Enabling APIs v1*, elaborated as part of Task 3.6 - Third Parties Enabling APIs, presents the detailed design for the SPHINX Application Programming Interface for Third Parties (S-API) component, as introduced in the SPHINX architecture (D2.6) [1]. The detailed design includes the identification and allocation of technical specifications defined in [1], further extended with specific requirements for the component's development, addressing its specificities, as well as interface specifications between S-API and the SPHINX components and third-parties.

This document provides the necessary basis for the technical implementation of the S-API component in SPHINX. It will be updated by a second version of the deliverable (D3.12 - Third Parties Enabling APIs v2) due for submission in June 2021, taking into consideration updates to the SPHINX architectural design and the ongoing development and integration work involving the S-API.

1.2 Structure of the deliverable

This document is structured as follows: section 2 presents the detailed design of S-API, which forms the main content of this deliverable. It starts by presenting a general overview of the S-API, with the identification of its positioning within the SPHINX Platform; it is followed by a presentation of S-API requirements, a set of sequence diagrams displaying the flow of interactions in key functions and the S-API interfaces. Section 3 delivers the conclusions of the work performed; and section 4 conveys the bibliographical references used in this document.

1.3 Relation to other WPs and Tasks

The elaboration of Task 3.6 - Third Parties Enabling APIs takes its main inputs from the work performed in WP2, specifically the definition of the SPHINX architecture [1] that provides system requirements and interfaces between components, developed in Task 2.5, also having as reference the SPHINX use cases (D2.4) [2] and user requirements (D2.5) [3]. Importantly, since S-API requires the interaction with SPHINX components providing services to third-parties, this work involved close interaction with the different WPs developing SPHINX components (i.e., WPs 3, 4 and 5) and WP 6 that deals with integration and deployment components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

2 SPHINX API for Third Parties

2.1 General Overview

The SPHINX Application Programming Interface for Third Parties (S-API) enables third-party solution providers to access and interact with the SPHINX Platform and its components. Subject to authentication, authorisation and using end-to-end encryption, S-API exposes advanced cyber security functionalities implemented by SPHINX components, such as device/application certification, threat registry notification and anomaly detection. The third-party interface specification for each component is presented in the respective component subsection. The S-API concept is presented in Figure 1.

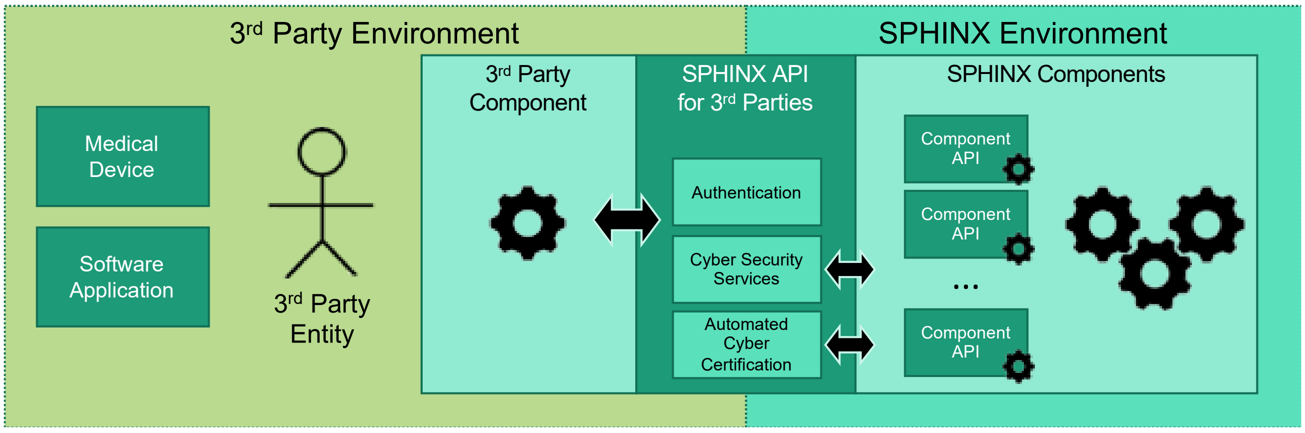


Figure 1: The S-API Concept

A particularly important feature in S-API consists on the delivery to third-parties of SPHINX device certification functionalities. Specifically, S-API may be used by medical devices manufacturers (constrained hardware running specialised software or firmware) and software services providers (specialised software applications and solutions) to access the SPHINX Sandbox and receive assurance that the device and services are SPHINX-compliant and certified, therefore becoming trusted assets in a SPHINX-secured information technology (IT) ecosystem.

2.2 S-API Architectural Overview

SPHINX provides an open API for third-parties, making its advanced cybersecurity functionalities and cyber certification capabilities available to them. This API provides a set of calls to invoke specific actions. The API calls should be well-documented and open to enable an easy integration by third parties. Via the API, third-parties will be able to discover and find which SPHINX services are available.

Aiming to maximise interoperability and ease of integration, interface specification follows OpenAPI [5] using the Swagger toolset [9]. Moreover, SPHINX supports the following type of interfaces with third-party components:

- JavaScript Object Notation (JSON) data format (RFC 8259 [4]);
- Web services based on the REpresentational State Transfer (REST) architecture, allowing devices to access services from the SPHINX Sandbox;
- OAuth2.0 as authorisation framework (RFC 8252 [6] and RFC 6750 [7]);

The architecture choice is designed to fully decouple the third-party component from SPHINX.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

The high-level architecture for the S-API is depicted in the next figure.

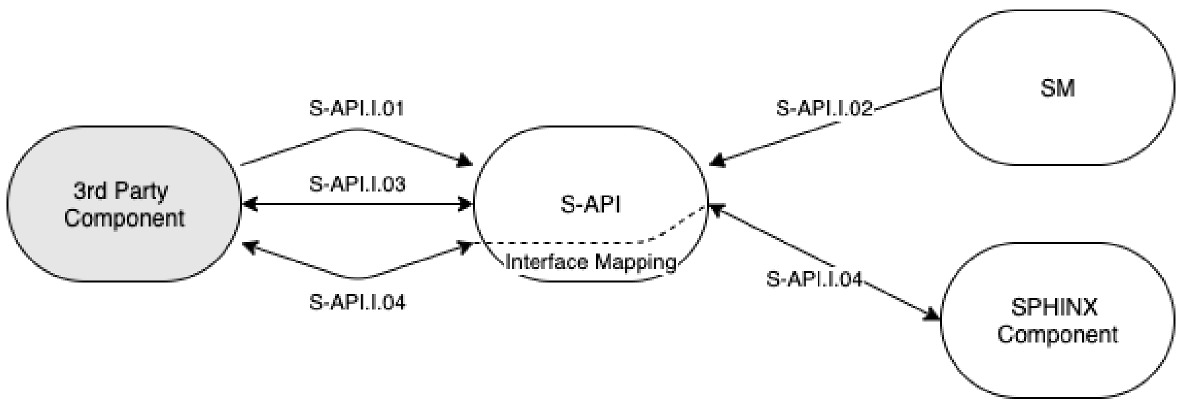


Figure 2: The S-API Architecture

The S-API component provides the following primary functions:

- **Administration and Management** of the S-API, allowing to manage third-party users, grant/revoke access and define service levels (e.g., permission to use a service, number of requests allowed) according to a third-party profile. This function is also used to define **subscription plans** for third-parties, each involving specific features and cost models (e.g., free, pay-per-use, monthly rate).
- **Third-Party Management Functions**, allowing third-party users to create and manage their account, providing information concerning their entity (personal, business or both) and select their appropriate **subscription plan**. Third-parties can also delete their account (and all associated data) at any time.
- **Third-Party Service Access Functions**, allowing third-parties to programmatically access functionalities provided by SPHINX services, including receiving notifications.
- **Third-Party SPHINX Certification Functions**, allowing access to the SPHINX Sandbox in order to validate and receive SPHINX compliance and certification reports concerning a third-party device and services.

The types of users defined for the S-API are:

- **S-API Administrator users**: refers to users that have administration roles to manage third-parties and their roles.
- **S-API Third-Party users**: refers to users representing third-parties to SPHINX that wish to access SPHINX services and functionalities via S-API.

For each of these functions, relevant technical specifications apply, extracted from deliverable *D2.6 - SPHINX Architecture v2* [1]. The requirements that are relevant for the adequate implementation of Task 3.6 results are identified in the following sections.

2.2.1.1 Allocated Requirements

The following requirements from D2.6 are allocated to the S-API.

Requirement	Description	Fulfilled	Comment
SYS-M-010	SPHINX shall be implemented following modular architecture principles.	Yes	Fully supported by the S-API.
SYS-M-020	SPHINX shall deliver a scalable platform.	Yes	Fully supported by the S-API.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Requirement	Description	Fulfilled	Comment
SYS-S-010	SPHINX shall provide data management functions to control sensitive data.	Yes	S-API benefits from SPHINX's data management functions.
SYS-S-020	SPHINX shall enforce secure management and storage of user credentials.	Yes	S-API benefits from SPHINX's secure management of user credentials.
SYS-S-030	SPHINX shall enable sessions management and re-authentication with single sign-on.	Yes	S-API benefits from SPHINX's secure management of user credentials.
SYS-S-070	SPHINX shall ensure that only authorised and authenticated users may access the system.	Yes	S-API benefits from SPHINX's authentication and authorisation (AAAC) mechanism.
S-API-F-010	SPHINX shall provide an open API to third parties enabling them to access SPHINX functionalities.	Yes	Fully supported by the S-API (cybersecurity discovery service).
S-API-S-020	SPHINX API access shall be authenticated and secure.	Yes	Fully supported by the S-API (authentication service).
S-API-S-030	SPHINX shall be able to manage credentials for API access.	Yes	Fully supported by the S-API (authentication service).
S-API-S-040	SPHINX shall be able to manage credential roles.	Yes	Fully supported by the S-API (authentication service).
S-API-S-050	SPHINX API shall provide a list of available services to a given third-party based on its role.	Yes	Fully supported by the S-API (cybersecurity discovery service).

Table 1: Relevant Technical Specifications for the S-API Service Components

2.2.2 S-API Service Component Requirements

This section specifies the requirements allocated to the S-API component. It starts with the general requirements and then presents the requirements per S-API sub-component.

2.2.2.1 S-API General Requirements

S-API shall provide open APIs enabling third-parties to access SPHINX functionalities.	
Requirement ID	S-API-DD-F-01
Requirement Type	Functional Specifications
Dependencies	S-API-F-010, STA-F-580 SYS-M-010, SYS-M-020
Description and Rationale	SPHINX services will be made available to third-parties via the S-API component. For this purpose, S-API will expose SPHINX services APIs to third-parties.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

SPHINX services providing third party access shall deliver well documented end-points and APIs.	
Requirement ID	S-API-DD-F-02
Requirement Type	Functional Specifications
Dependencies	STA-F-580
Description and Rationale	SPHINX services providing third-party access shall deliver well defined end-points and well documented interfaces regarding their APIs. The interface specification shall follow OpenAPI [5], using the Swagger toolset [9].

S-API shall be granted permissions to access all SPHINX services and interfaces providing interfaces to third-parties.	
Requirement ID	S-API-DD-S-01
Requirement Type	Security Specifications
Dependencies	STA-F-580
Description and Rationale	S-API access to SPHINX will be subject to authentication and authorisation. Upon successful authentication, S-API shall be granted access to all SPHINX services and interfaces.

2.2.2.2 S-API Administration and Management (AM)

S-API AM deals with functions related with housekeeping and third-party access in SPHINX, including management of third-party users and setting their access rights. S-API AM are not available to third-parties.

The following requirements are specified for the S-API AM.

S-API AM access shall require secure authentication.	
Requirement ID	S-API-DD-S-10
Requirement Type	Security Specifications
Dependencies	S-API-S-020 SYS-S-020, SYS-S-070
Description and Rationale	Specific user accounts within S-API (i.e., S-API administrators) will be created with permissions to access the S-API AM. Only after successful authentication, a S-API administrator can access the S-API AM functions. Authentication shall use end-to-end encryption to ensure confidentiality.

S-API AM shall allow the creation and management of third-party users.	
Requirement ID	S-API-DD-F-10
Requirement Type	Functional Specifications
Dependencies	S-API-S-030, S-API-S-040
Description and Rationale	Third-party users can be created by S-API administrators (provided a third-party has granted authorisation). A S-API administrator can also revoke (i.e., temporarily disable) access to a third-party and remove a third-party from SPHINX (thus deleting all its data).

S-API AM shall allow defining access permissions and usage profiles for third-party users.	
Requirement ID	S-API-DD-F-11
Requirement Type	Functional Specifications
Dependencies	STA-F-590
Description and Rationale	A third-party shall be granted a role based on its selected profile and preferences. The third-party role establishes which services it can access (as well as their interfaces), usage profiles (e.g., limitation in number of service requests) and added-value services (e.g., email push-notifications). The role is used to associate a third-party to specific subscription plans. A third-party role shall be subjected to approval by the S-API AM.

S-API AM shall provide an access token allowing third-party's components to access the SPHINX Third-Party API.	
Requirement ID	S-API-DD-S-11
Requirement Type	Security Specifications
Dependencies	S-API-S-030
Description and Rationale	The S-API AM shall provide for each third-party an authorisation token that will allow the third-party to access the SPHINX Third-Party API. The token is used to replace the need to use user credentials (i.e., username and password) when accessing the APIs. Access tokens are managed by the S-API AM and can be revoked at any time.

2.2.2.3 S-API Third-Party Management Functions

The Third-Party Management Functions (TPMF) allow third-party users to manage their accounts (including create, modify and delete their data) and select their intended role for access permissions and usage profiles.

The following requirements are specified for the S-API TPMF.

Third-party access shall require secure authentication.	
Requirement ID	S-API-DD-S-20
Requirement Type	Security Specifications
Dependencies	S-API-S-020 SYS-S-020, SYS-S-070
Description and Rationale	In order to access SPHINX functionalities, third-parties are required to be authenticated by the S-API component. Authentication shall use end-to-end encryption to ensure confidentiality.

S-API shall allow a third-party to create a third-party user account.	
Requirement ID	S-API-DD-F-20
Requirement Type	Functional Specifications
Dependencies	S-API-S-030, S-API-S-040
Description and Rationale	A third-party can use S-API to create a user account. S-API shall grant the third-party a default role (as defined by the S-API administrator).

S-API shall allow a third-party to manage its third-party account.	
Requirement ID	S-API-DD-F-21
Requirement Type	Functional Specifications
Dependencies	S-API-S-030, S-API-S-040
Description and Rationale	A third-party can use the S-API to manage its user account. This involves: - Edit the third-party's information (e.g., company's name, address); - Edit the contact's information (e.g., person's name); - Change the contact email and password; - Delete the account (thus removing all stored third-party data).

S-API shall allow a third-party to request a change in role.	
Requirement ID	S-API-DD-F-22
Requirement Type	Functional Specifications
Dependencies	STA-F-590, S-API-DD-F-11
Description and Rationale	A third-party can use the S-API to visualise its current role and can request a role change. The request will be subject to approval by the S-API Administrator.

S-API shall allow a third-party to recover its user credentials.	
Requirement ID	S-API-DD-S-21
Requirement Type	Security Specifications
Dependencies	S-API-S-030
Description and Rationale	In case a third-party loses its credentials, it can use the S-API to recover them. The S-API will send an email to the third-party contact email containing the recovery instructions. Furthermore, an information email will also be sent to (1) a third-party registered email and (2) S-API Administrators.

2.2.2.4 S-API Third-Party Service Access Functions

The S-API Third-Party Service Access Functions (TPSAF) allow authorised third-parties to access the SPHINX Services APIs provided via S-API. The API is meant to be accessed programmatically by third-parties' components.

The following requirements are specified for the S-API TPSAF.

A third-party shall be able to discover and retrieve SPHINX services.	
Requirement ID	S-API-DD-F-30
Requirement Type	Functional Specifications
Dependencies	S-API-S-050 S-API-DD-F-11
Description and Rationale	Third-parties can use the S-API to discover and visualise a list of SPHINX services. The list shall contain, for each service, the service's description, configuration and the interface specification to third-parties (JSON structured file). Access to services will be granted or denied based on the third-party's role, as defined by the S-API AM.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

A third-party component shall be able to programmatically access a SPHINX service via S-API.	
Requirement ID	S-API-DD-F-31
Requirement Type	Functional Specifications
Dependencies	STA-F-580
Description and Rationale	S-API shall allow a third-party user to programmatically access a SPHINX service functionality. The applicable SPHINX service must have an API enabled for this purpose. S-API shall provide a mapping between third-party requests and SPHINX services. For loosely-coupling components and to maximise interoperability, the APIs shall follow the OpenAPI specifications, use HyperText Transfer Protocol (HTTP) REST and, unless deemed unfeasible, data in JSON format.

S-API shall support on-demand (synchronous) and event-based requests.	
Requirement ID	S-API-DD-F-32
Requirement Type	Functional Specifications
Dependencies	STA-F-580
Description and Rationale	Third-parties can issue the following type of requests to the S-API: • On-demand (synchronous) requests, where a third-party issues a request to a SPHINX service waiting for its response (or timeout due to an error). For example, this is the case of a HTTP GET request. • Notification-based request, where a third-party issues a request to a SPHINX service and receives its response by means of a notification. For this, a third-party is required to subscribe to events of interest.

Third-party's components access to APIs shall require secure authentication.	
Requirement ID	S-API-DD-S-30
Requirement Type	Security Specifications
Dependencies	S-API-S-020, S-API-S-030 SYS-S-070 S-API-DD-S-11
Description and Rationale	In order to access SPHINX functionalities via S-API, third-parties' components are required to provide authentication credentials, by using the access token provided by the S-API. Authentication shall use end-to-end encryption to ensure confidentiality.

2.2.2.5 *Third-Party SPHINX Certification Functions*

The S-API Third-Party SPHINX Certification Functions (TPCF) shall enable programmatic access to the SPHINX Sandbox. Specifically, S-API may be used by medical devices manufacturers (constrained hardware running specialised software or firmware) and software services providers (specialised software applications and solutions) to access the SPHINX Sandbox and receive assurance that the device and services are SPHINX-compliant and certified, therefore becoming trusted assets in a SPHINX-secured IT ecosystem.

If the third-party component is considered untrusted, the Sandbox is invocated to isolate the process conducting the appropriate auditing to certify it. In this way, certification is performed in a controlled environment, not disrupting normal operations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

However, the certification process can also be requested and executed in already deployed systems or assets (i.e., assumed trusted).

The certification process takes two steps:

- In the first, named "*certification request*", the third-party issues the certification request to the Sandbox providing the necessary information concerning the module to be certified. If the module is considered to be trusted (already deployed in operational environment), no isolated environment will be created.
- In the second, named "*certification result*", the third-party receives the certification report, issued by the Sandbox's Automated Cybersecurity Certification, via a S-API message event, that contains the results of the certification process

S-API shall allow third-parties to issue certification requests.	
Requirement ID	S-API-DD-F-40
Requirement Type	Functional Specifications
Dependencies	S-API-F-010
Description and Rationale	A third-party can use S-API to issue a certification request to the Sandbox, also providing the necessary information concerning the module to be certified.

S-API shall convey certification results to third-parties.	
Requirement ID	S-API-DD-F-41
Requirement Type	Functional Specifications
Dependencies	S-API-F-010 S-API-DD-F-32
Description and Rationale	A third-party receives a certification report, issued by the Sandbox's Automated Cybersecurity Certification, via a S-API message event.

Third-party's components access to SPHINX certification functions shall require secure authentication and proper role.	
Requirement ID	S-API-DD-S-40
Requirement Type	Security Specifications
Dependencies	S-API-S-030 SYS-S-070 S-API-DD-S-11
Description and Rationale	In order to access the S-API TPCF, third-parties' components are required to provide authentication credentials, by using the access token provided by the S-API. Authentication shall use end-to-end encryption to ensure confidentiality. Access to the S-API TPCF is granted or rejected based on the third-party's role.

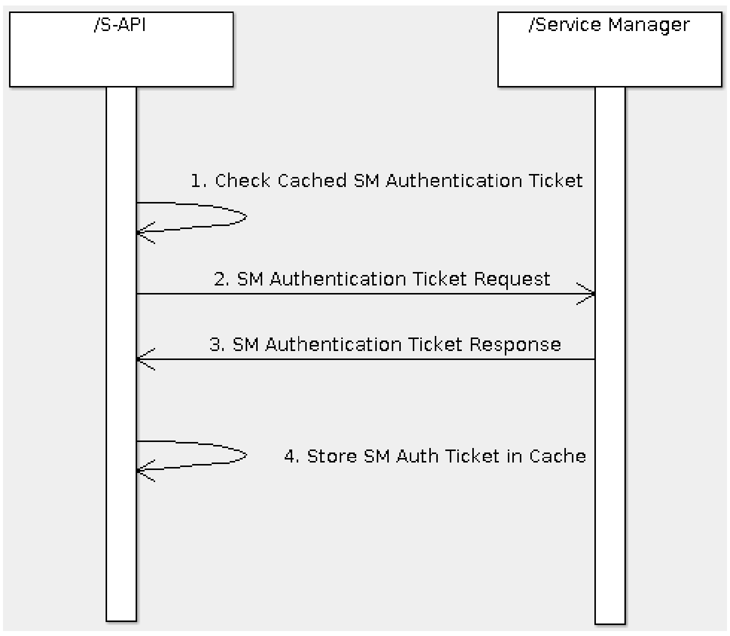
2.3 S-API Sequence Diagrams

In this section, sequence diagrams related with specific functions of S-API are presented. It details the process and the sub-components involved in each of the functional processes executed by the S-API component. Its goal is to present a high-level interaction between sub-components, involving a successful sequence of operations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

2.3.1 S-API Authentication in SPHINX

The following sequence diagram depicts the registration process for S-API in SPHINX. The process requires that the username/password credentials have been created by the Service Manager (SM).


/S-API
/Service Manager
1. Check Cached SM Authentication Ticket
2. SM Authentication Ticket Request
3. SM Authentication Ticket Response
4. Store SM Auth Ticket in Cache

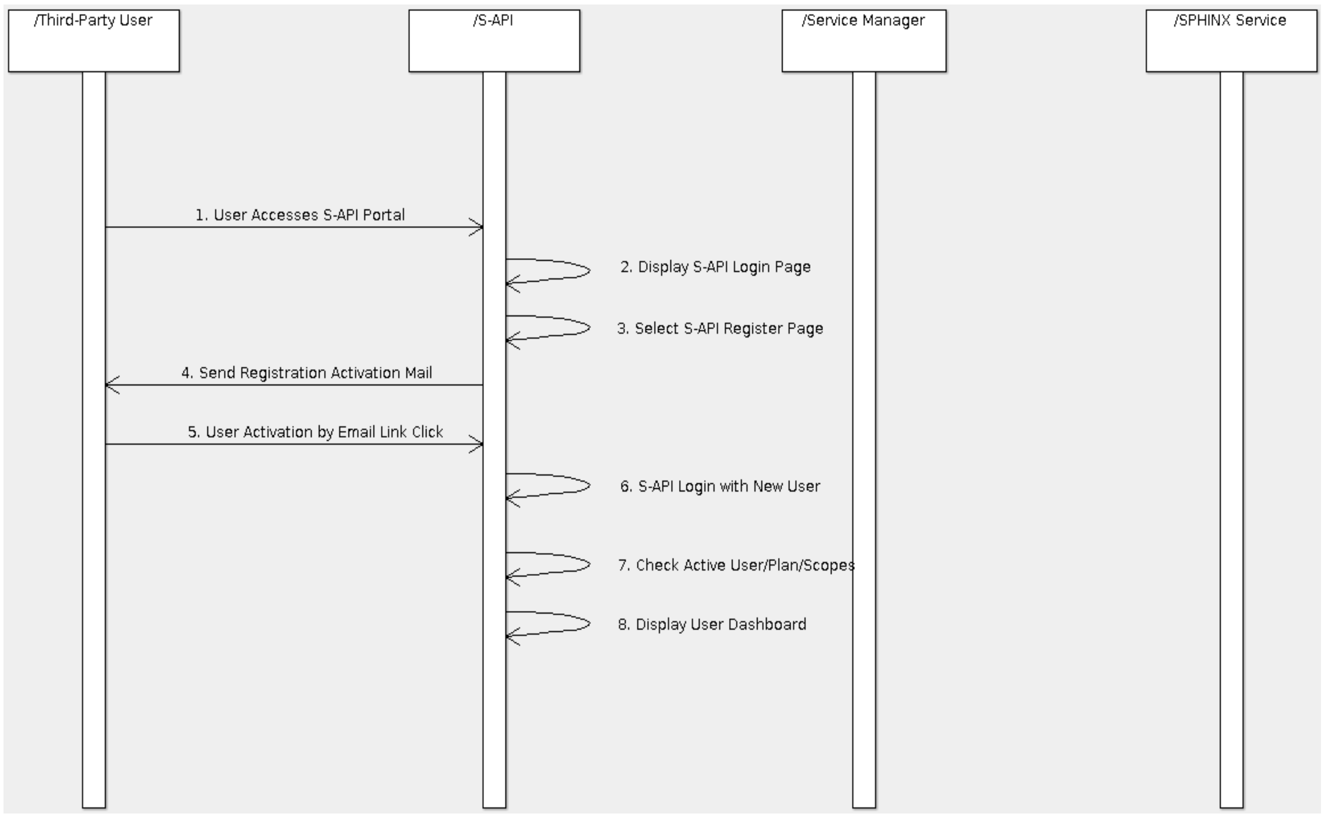
The sequence of actions is the following:

1. Check Cached SM Authentication Ticket	S-API -> S-API	• Check if there is a valid SM authentication ticket stored locally in cache. • If such ticket is valid, this sequence is stopped (no need to run 2, 3 and 4).
2. SM Authentication Ticket Request	S-API -> Service Manager	• If not previously cached on S-API, S-API requests an authentication ticket to the SM. A username/password credentials pair is provided as parameters.
3. SM Authentication Ticket Response	Service Manager -> S-API	• If the authentication is successful, the SM provides the authentication ticket code as response. • The ticket locally cached by the S-API is reused in following SPHINX service calls.
4. Store SM Authentication Ticket in Cache	S-API -> S-API	• Store the retrieved token in a local cache, along with any Time-To-Leave (TTL) information to avoid repeating this process every time a third-party request is made.

2.3.2 Registration

The following sequence diagram depicts the registration process for third-parties. Once completed, the third-party becomes registered in the S-API component. Note that third-party accounts are fully managed by S-API.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.


/Third-Party User
/S-API
/Service Manager
/SPHINX Service
1. User Accesses S-API Portal
2. Display S-API Login Page
3. Select S-API Register Page
4. Send Registration Activation Mail
5. User Activation by Email Link Click
6. S-API Login with New User
7. Check Active User/Plan/Scopes
8. Display User Dashboard

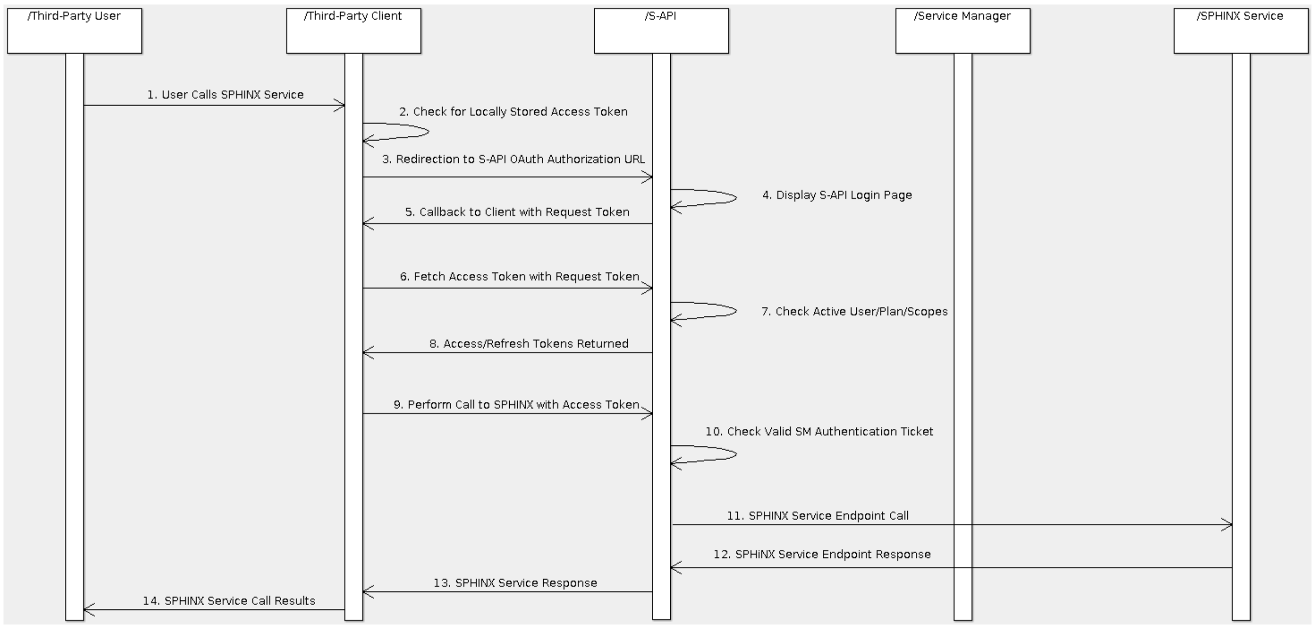
The sequence of actions is the following:

1. User Accesses the S-API Portal	Third-Party User -> S-API	• The third-party user uses a browser to open the S-API homepage.
2. Display S-API Login Page	S-API -> S-API	• A login window is presented with username/password fields. If login is selected, step 7 is executed. • Additionally, a Register New User option is visible. If selected, step 3 is executed.
3. Display S-API Register Page	S-API -> S-API	• A registration window is presented with username/password and General Data Protection Regulation (GDPR)-related settings. • Terms and conditions are available for the third-party user to read, acknowledge and accept them.
4. Send Registration Activation Mail	S-API -> Third-Party User	• S-API triggers an e-mail notification including a link to activate the new third-party user that is disabled at the moment.
5. User Activation by Email Link Click	Third-Party User -> S-API	• After the third-party user clicks the link on the email, the third-party user is redirected again to the Login page.
6. S-API Login with New User	S-API -> S-API	• S-API login window is presented again for the third-party user to authenticate with its credentials (username/password).
7. Check Active User/Plan/Scopes	S-API -> S-API	• S-API evaluates if the third-party user is enabled on the system and if he/she has a valid plan that allows him/her to use the

		system.
8. Display User Dashboard	S-API -> Third-Party User	• After the login/plan validations have been checked, the third-party user enters the S-API Portal and is redirected to the third-party user's dashboard page.

2.3.3 Login

The following sequence diagram depicts the login process for third-parties. For completeness purposes, it also includes a request to a SPHINX service performed via S-API and the SM.


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Service
1. User Calls SPHINX Service
2. Check for Locally Stored Access Token
3. Redirection to S-API OAuth Authorization URL
4. Display S-API Login Page
5. Callback to Client with Request Token
6. Fetch Access Token with Request Token
7. Check Active User/Plan/Scopes
8. Access/Refresh Tokens Returned
9. Perform Call to SPHINX with Access Token
10. Check Valid SM Authentication Ticket
11. SPHINX Service Endpoint Call
12. SPHINX Service Endpoint Response
13. SPHINX Service Response
14. SPHINX Service Call Results

The sequence of actions is the following:

1. User Calls SPHINX Service	Third-Party User -> Third-Party Client	• The third-party uses a third-party client to request a call to a SPHINX service.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• The third-party client tries to use a cached version of an access token, if available. • If the cached access token does not exist, the third-party client is redirected to S-API Authorisation link with step 3. • Otherwise, move to step 6.
3. Redirection to S-API OAuth Authorisation URL	Third-Party Client -> S-API	• The third-party client is redirected to the S-API OAuth authorisation endpoint, providing a custom redirect Uniform Resource Locator (URL) as call back or using the default S-API redirect URL for the application. • If the third-party client has not logged in on S-API previously, he/she/it is redirected to the S-API Login page (step 4). • If he/she/it has logged in previously, an authorisation page is displayed to ask for the user's approval. Depending on OAuth settings on S-API, this authorisation flow might be implicit.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4. Display S-API Login Page	S-API -> S-API	• A login window is presented with login/password fields. • The third-party user enters credentials for an existing S-API user.
5. Call back to Client with Request Token	S-API -> Third-Party Client	• The third-party client is notified via the call back provided on step 3. • This call back contains a temporary request token to be used on step 10 to request the final access token.
6. Fetch Access Token with Request Token	Third-Party Client -> S-API	• The S-API OAuth access token request endpoint is invoked. • This access token may or might not be returned, depending if the third-party user has a proper role, plan or if he/she/it is disabled via the S-API back-office.
7. Check Active User/Plan/Scopes	S-API -> S-API	• S-API evaluates if the third-party user is enabled on the system and if he/she/it has a valid plan that allows him/her/it to use the system.
8. Access/Refresh Tokens Returned	S-API -> Third-Party Client	• If user requirements are fulfilled, the call returns a new access token. • A refresh token is also returned, which can be used to renew manually/automatically the access token if expired via OAuth token refresh endpoint.
9. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the obtained access token. • The access token might be provided via URL or HTTP header, depending on the S-API implementation.
10. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the “Service Manager (SM) Login” process is executed (see 2.3.1).
11. SPHINX Service Endpoint Call	S-API -> SPHINX Service	• S-API uses previously cached SM authentication ticket to invoke the SPHINX service endpoint directly.
12. SPHINX Service Endpoint Response	SPHINX Service -> S-API	• The SPHINX Service executes the call and returns a proper response for the call to S-API.
13. SPHINX Service Response	S-API -> Third-Party Client	• The S-API service conveys the SPHINX service response to the third-party client.
14. SPHINX Service Call Results	Third-Party Client -> Third-Party User	• The SPHINX service call results are properly processed by the third-party client application and presented to the third-party user.

2.3.4 Service List Request

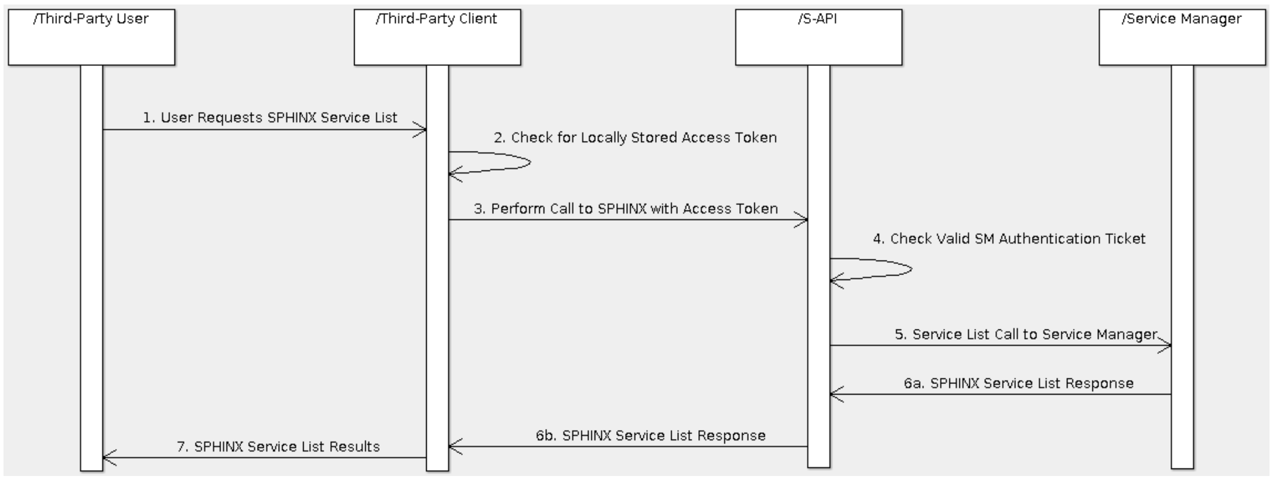
The following sequence diagram depicts a third-party request for the list of SPHINX third-party services.

For the request to be successful:

- S-API needs to be authenticated in SPHINX (access token provided by SPHINX);
- The third-party user needs a valid access token (access token provided by S-API).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Note that all third-party users are authorised by S-API to retrieve the service list request from SPHINX.


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
1. User Requests SPHINX Service List
2. Check for Locally Stored Access Token
3. Perform Call to SPHINX with Access Token
4. Check Valid SM Authentication Ticket
5. Service List Call to Service Manager
6a. SPHINX Service List Response
6b. SPHINX Service List Response
7. SPHINX Service List Results

The sequence of actions is the following:

1. User Calls SPHINX Service	Third-Party User -> Third-Party Client	• The third-party user uses a third-party client to request the list of SPHINX Services.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• The third-party client uses the cached version of an access token. • Note it is assumed that the login flow has occurred previously.
3. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the cached access token. • The access token might be provided via URL or HTTP header, depending on the S-API implementation.
4. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the "Service Manager (SM) Login" process is executed (see 2.3.1).
5. Service List Call to Service Manager	S-API -> Service Manager	• The SM provides via a specialised endpoint or function a list of all the SPHINX services available for S-API.
6. SPHINX Service List Response	Service Manager -> Third-Party Client	• The list of available services is parsed by S-API, cached locally and is propagated towards the third-party client.
7. SPHINX Service List Results	Third-Party Client -> Third-Party User	• The SPHINX service list is conveyed and presented to the third-party user.

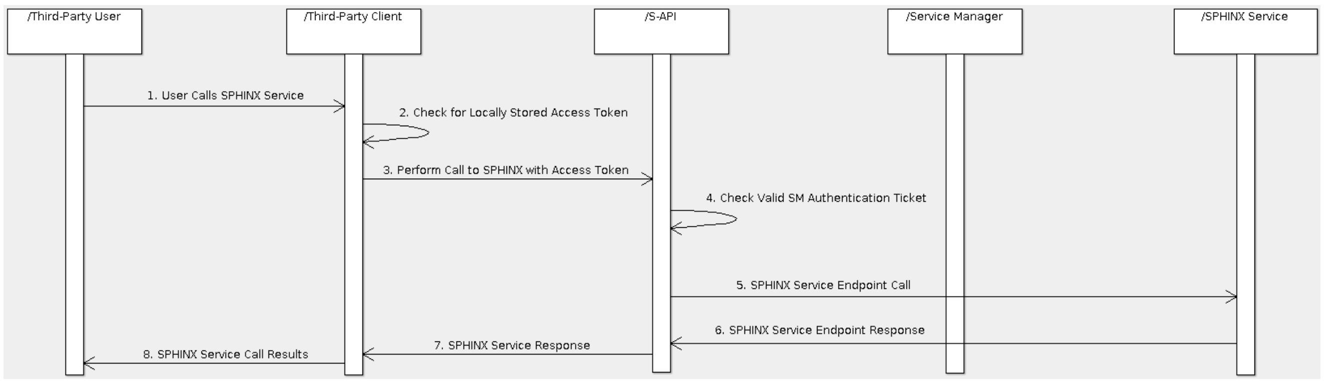
2.3.5 Service Request

The following sequence diagram depicts a third-party request for a particular SPHINX service.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

For the request to be successful:

- S-API needs to be authenticated in SPHINX (access token provided by SPHINX);
- The third-party user needs a valid access token (access token provided by S-API);
- The third-party user is authorised (by S-API) to perform the service request.


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Service
1. User Calls SPHINX Service
2. Check for Locally Stored Access Token
3. Perform Call to SPHINX with Access Token
4. Check Valid SM Authentication Ticket
5. SPHINX Service Endpoint Call
6. SPHINX Service Endpoint Response
7. SPHINX Service Response
8. SPHINX Service Call Results

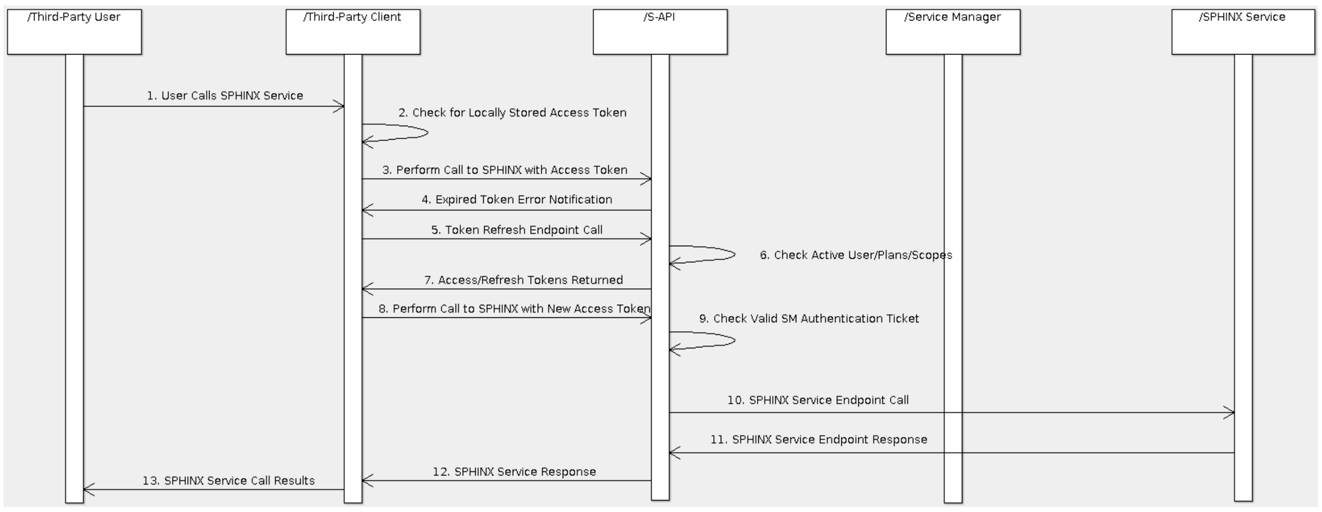
The sequence of actions is the following:

1. User Calls SPHINX Service	Third-Party User -> Third-Party Client	• The third-party uses a third-party client to request a call to a SPHINX service.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• Third-party client uses the cached version of the access token. • Note it is assumed that the login flow has occurred previously.
3. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the obtained access token. • S-API verifies that the third-party is authorised to perform the request. • The access token might be provided via URL or HTTP header, depending on the S-API implementation.
4. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the “Service Manager (SM) Login” process is executed (see 2.3.1).
5. SPHINX Service Endpoint Call	S-API -> SPHINX Service	• S-API uses previously cached SM authentication ticket to invoke the SPHINX service endpoint directly.
6. SPHINX Service Endpoint Response	SPHINX Service -> S-API	• SPHINX Service executes the call and returns a response (herein, a synchronous request is assumed).
7. SPHINX Service Response	S-API -> Third-Party Client	• The S-API service conveys the received response to the third-party client.
8. SPHINX Service Call Results	Third-Party Client -> Third-Party User	• The result of the SPHINX service call is conveyed and presented to the third-party user.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

2.3.6 Access Token Refresh

The following sequence diagram depicts a third-party request for a particular SPHINX service, when the user's access token has expired and needs to be refreshed.


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Service
1. User Calls SPHINX Service
2. Check for Locally Stored Access Token
3. Perform Call to SPHINX with Access Token
4. Expired Token Error Notification
5. Token Refresh Endpoint Call
6. Check Active User/Plans/Scopes
7. Access/Refresh Tokens Returned
8. Perform Call to SPHINX with New Access Token
9. Check Valid SM Authentication Ticket
10. SPHINX Service Endpoint Call
11. SPHINX Service Endpoint Response
12. SPHINX Service Response
13. SPHINX Service Call Results

The sequence of actions is the following:

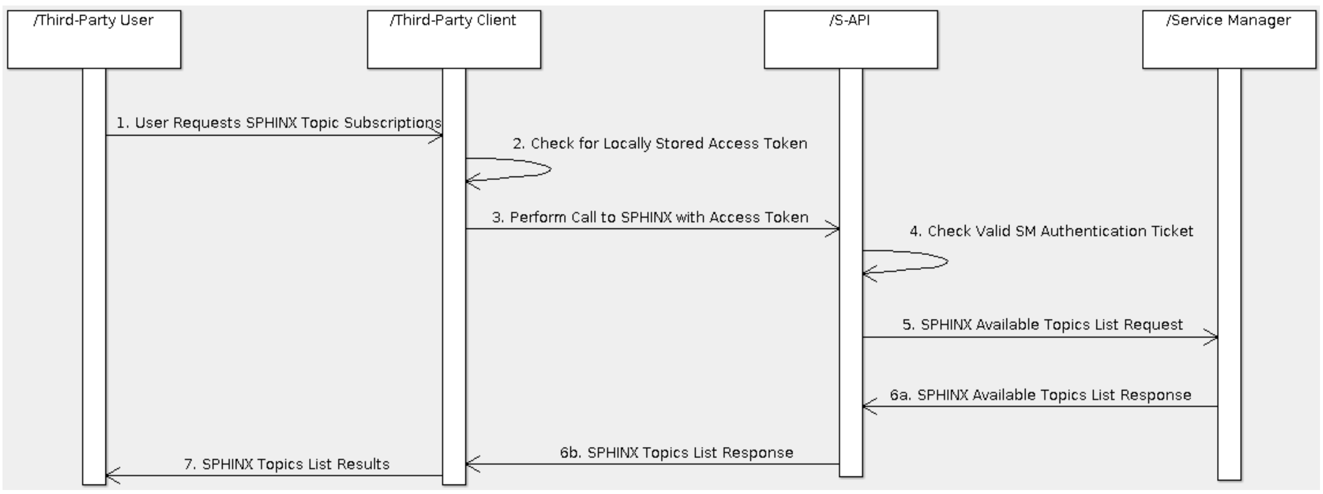
1. User Calls SPHINX Service	Third-Party User -> Third-Party Client	• The third-party uses a third-party client to request a call to a SPHINX service.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• Third-party client uses the cached version of the access token. • Note it is assumed that the login flow has occurred previously.
3. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• Third-party client tries to perform a generic service/service list call with an expired access token.
4. Expired Token Error Notification	S-API -> Third-Party Client	• The S-API returns an OAuth error notifying that the token has expired. This triggers a token refresh call (manually or automatically, depending on OAuth client library), according to step 5.
5. Token Refresh Endpoint Call	Third-Party Client -> S-API	• This step (manual or automatic, depending on the OAuth client) invokes the S-API OAuth token refresh endpoint, providing the previously retrieved refresh token.
6. Check Active User/Plan/Scopes	S-API -> S-API	• S-API evaluates if the user is enabled on the system and if he/she has a valid plan which allows him/her to use the system.
7. Access/Refresh Tokens Returned	S-API -> Third-Party Client	• If user requirements are fulfilled, the call returns a new access token. • A refresh token is also returned, which can be used to renew manually/automatically the access token if expired, via OAuth token refresh endpoint.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

8. Perform Call to SPHINX with New Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the obtained access token. • S-API verifies that the third-party is authorised to perform the request. • The access token might be provided via URL or HTTP header, depending on the S-API implementation
9. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the "Service Manager (SM) Login" process is executed (see 2.3.1).
10. SPHINX Service Endpoint Call	S-API -> SPHINX Service	• S-API uses previously cached SM authentication ticket to invoke the SPHINX service endpoint directly.
11. SPHINX Service Endpoint Response	SPHINX Service -> S-API	• SPHINX Service executes the call and returns a proper response for the call to the S-API.
12. SPHINX Service Response	S-API -> Third-Party Client	• The S-API service conveys the SPHINX service response to the third-party client.
13. SPHINX Service Call Results	Third-Party Client -> Third-Party User	• The SPHINX service call results are properly processed by the third-party client application and presented to the third-party user.

2.3.7 Retrieve a SPHINX Service's List of Topics

The following sequence diagram depicts a third-party request to receive the list of topics a SPHINX service uses to publish messages.

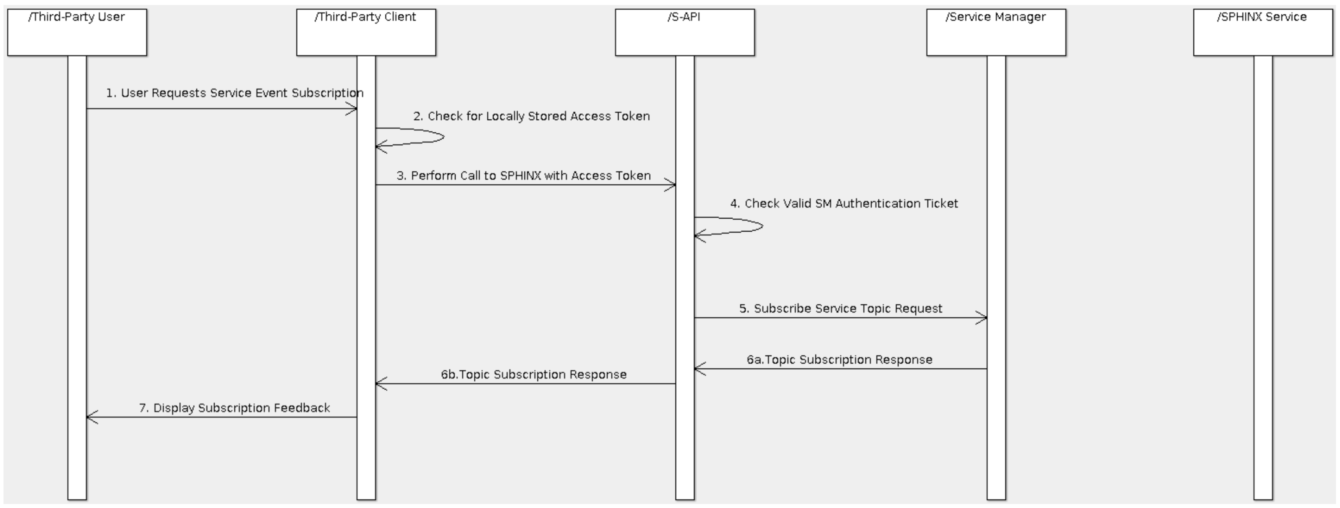

/Third-Party User
/Third-Party Client
/S-API
/Service Manager
1. User Requests SPHINX Topic Subscriptions
2. Check for Locally Stored Access Token
3. Perform Call to SPHINX with Access Token
4. Check Valid SM Authentication Ticket
5. SPHINX Available Topics List Request
6a. SPHINX Available Topics List Response
6b. SPHINX Topics List Response
7. SPHINX Topics List Results

The sequence of actions is the following:

1. User Requests SPHINX Topic Subscriptions	Third-Party User -> Third-Party Client	• The third-party user uses a third-party client to retrieve the list of all SPHINX topics available for subscription.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• The third-party client uses the cached version of an access token. • Note it is assumed that the login flow has occurred previously.
3. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the cached access token. • S-API verifies that the third-party is authorised to perform the request. • The access token might be provided via URL or HTTP header, depending on S-API implementation.
4. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the "Service Manager (SM) Login" process is executed.
5. SPHINX Available Topics List Request	S-API -> Service Manager	• Using a valid authentication ticket, the list of all SPHINX topics available for subscription is requested.
6. SPHINX Available Topics List Request	Service Manager -> S-API	• The SM sends the list of topics to S-API, which is locally cached. • S-API sends the list to the third-party client.
7. SPHINX Topics List Results	Third-Party Client -> Third-Party User	• The list of all SPHINX topics available for subscription is conveyed and presented to the third-party user.

2.3.8 Event Topic Subscription

The following sequence diagram depicts a third-party subscribing a topic.


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Service
1. User Requests Service Event Subscription
2. Check for Locally Stored Access Token
3. Perform Call to SPHINX with Access Token
4. Check Valid SM Authentication Ticket
5. Subscribe Service Topic Request
6a.Topic Subscription Response
6b.Topic Subscription Response
7. Display Subscription Feedback

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

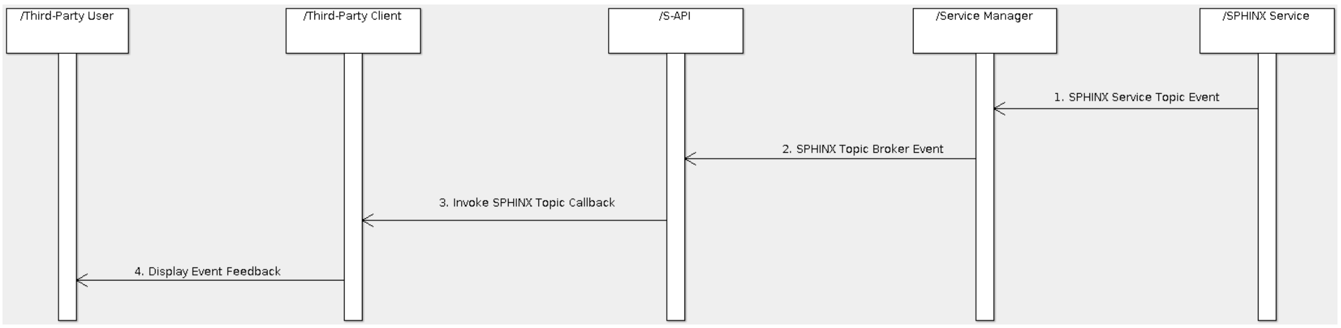
The sequence of actions is the following:

1. User Requests Service Event Subscription	Third-Party User -> Third-Party Client	• The third-party user uses a third-party client to subscribe to a particular topic.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• The third-party client uses the cached version of an access token. • Note it is assumed that the login flow has occurred previously.
3. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the cached access token. • S-API verifies that the third-party is authorised to perform the request. • The access token might be provided via URL or HTTP header, depending on the S-API implementation
4. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the “Service Manager (SM) Login” process is executed (see 2.3.1).
5. Subscribe Service Topic Request	S-API -> Service Manager	• Using a valid authentication ticket, S-API subscribes to the particular topic using the SM.
6. Topic Subscription Response	Service Manager -> Third-Party Client	• The SM returns the result of the subscription (success).
7. Display Subscription Feedback	Third-Party Client -> Third-Party User	• The third-party client returns the result of the subscription (success).

2.3.9 Message Event Associated with a Subscribed Topic

The following sequence diagram depicts a third-party receiving a message associated with a topic.

For this to be successful, the third-party must have successfully subscribed to the respective topic (see 2.3.8).


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Service
1. SPHINX Service Topic Event
2. SPHINX Topic Broker Event
3. Invoke SPHINX Topic Callback
4. Display Event Feedback

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

The sequence of actions is the following:

1. SPHINX Service Topic Event	SPHINX Service -> Service Manager	• As a result of the occurrence of a specific event, a SPHINX service publishes a message to a predefined topic in the SM.
2. SPHINX Topic Broker Event	Service Manager -> S-API	• The SM notifies all consumers that subscribed to the specific topic. The notifications are conveyed through S-API.
3. Invoke SPHINX Topic Call back	S-API -> Third-Party Client	• S-API conveys all notifications to third-party clients that subscribed to the respective topics.
4. Display Event Feedback	Third-Party Client -> Third-Party User	• The third-party client presents the message related with the notification event to the third-party user.

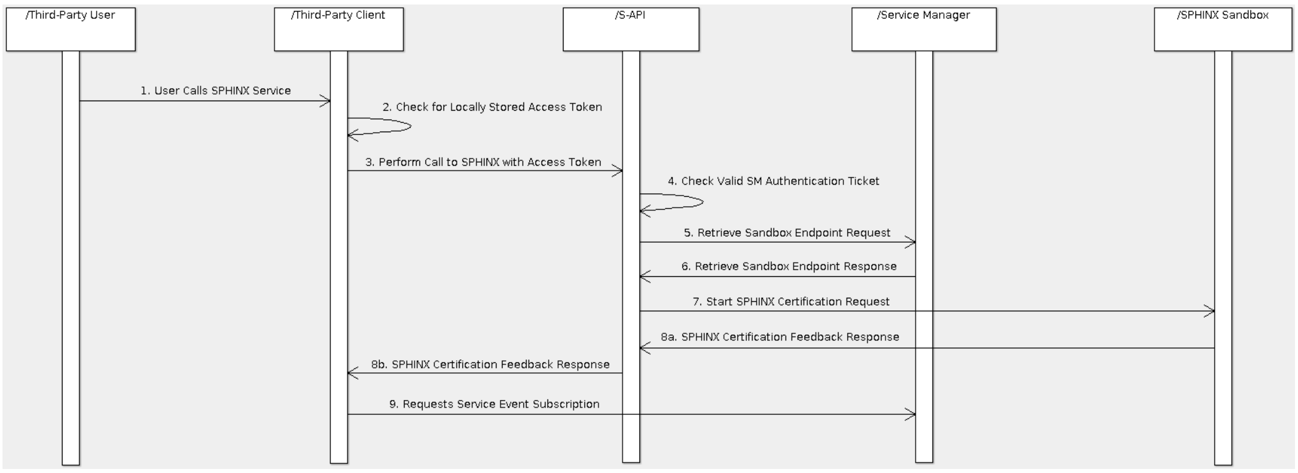
2.3.10 SPHINX Certification Workflow

The following sequence diagram depicts a third-party issuing a certification request to SPHINX.

The certification takes two steps:

- In the first, named "*certification request*", the third-party issues the certification request to the Sandbox providing the necessary information concerning the module to be certified.
- In the second, named "*certification result*", the third-party receives the certification report from the Automated Cybersecurity Certification, via a message event, that contains the results of the certification process.

First Step: Certification Request

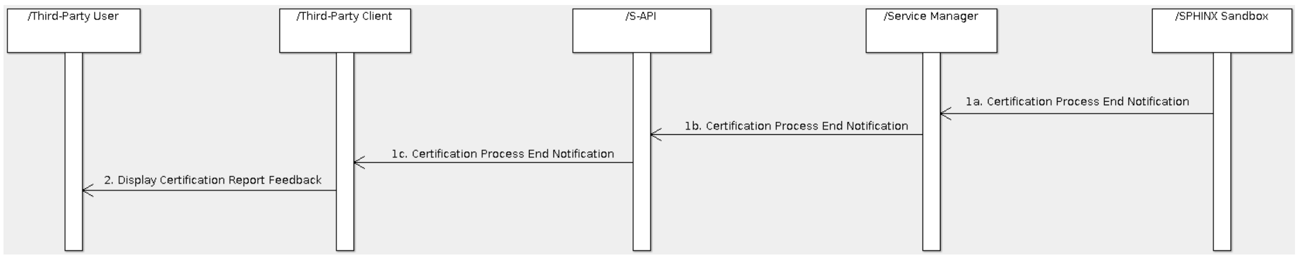

/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Sandbox
1. User Calls SPHINX Service
2. Check for Locally Stored Access Token
3. Perform Call to SPHINX with Access Token
4. Check Valid SM Authentication Ticket
5. Retrieve Sandbox Endpoint Request
6. Retrieve Sandbox Endpoint Response
7. Start SPHINX Certification Request
8a. SPHINX Certification Feedback Response
8b. SPHINX Certification Feedback Response
9. Requests Service Event Subscription

The sequence of actions is the following:

1. User Calls SPHINX Service	Third-Party User -> Third-Party Client	• The third-party user uses a third-party client to issue a certification request.
2. Check for Locally Stored Access Token	Third-Party Client -> Third-Party Client	• Third-party client uses the cached version of an access token.

		• Note it is assumed that the login flow has occurred previously.
3. Perform Call to SPHINX with Access Token	Third-Party Client -> S-API	• The third-party invokes the SPHINX service call, using the cached access token. • S-API verifies that the third-party is authorised to perform the request. • The access token might be provided via URL or HTTP header, depending on the S-API implementation.
4. Check Valid SM Authentication Ticket	S-API	• Check if there is a valid SM authentication ticket cached locally. • If no valid local ticket exists, the “Service Manager (SM) Login” process is executed (see 2.3.1).
5. Retrieve Sandbox Endpoint Request	S-API -> Service Manager	• If not available on the S-API local cache, fetch a reference for the SPHINX Sandbox service in order to retrieve the endpoint to start a certification process.
6. Retrieve Sandbox Endpoint Request	Service Manager -> S-API	• The SM returns the necessary information about the service responsible for certification requests.
7. Start SPHINX Certification Request	S-API -> SPHINX Sandbox	• The certification process is triggered, providing to the Sandbox component the required information to execute the certification process (e.g., IP address, network ports, database/application name).
8. SPHINX Certification Feedback Response	SPHINX Sandbox -> S-API -> Third-Party Client	• The Sandbox component returns: success or error indication; and, if successful, the corresponding topic ID to which the certification report is published (see next diagram). • S-API conveys the results of the certification request to the third-party client (i.e., topic ID). • The certification response is asynchronous because the certification process itself may take a while.
9. Requests Service Event Subscription	Third-Party Client -> S-API -> SM	• The third-party client subscribes to the topic ID.

Second Step: Certification Result


/Third-Party User
/Third-Party Client
/S-API
/Service Manager
/SPHINX Sandbox
1a. Certification Process End Notification
1b. Certification Process End Notification
1c. Certification Process End Notification
2. Display Certification Report Feedback

1. Certification Process End Notification	SPHINX Sandbox -> Third-Party Client	• When the certification process is finished, the third-party client is notified, via the subscribed topic to the SM.
2. Display Certification Report Feedback	Third-Party Client -> Third-Party User	• The third-party client shows the certification report to the user.

2.4 S-API Interfaces

This section lists the S-API interfaces considering two different types of interactions:

- S-API management and administration functions;
- Third-party access to SPHINX services via S-API.

Note that the SPHINX components' detailed interfaces are documented in the project's GitLab repository [8]. See also D2.6 [1] for a functional description of the SPHINX interfaces for third-parties.

2.4.1 S-API Interfaces for Management and Administration

For the integration with third-party components (equipment, software applications and services), SPHINX provides a set of simple RESTful Web interfaces that facilitates the third-parties' authentication with the SPHINX System, the third-parties' authorised access to the available SPHINX cybersecurity services and the third-parties' certification of their components, verifying their compliance to the SPHINX cybersecurity standards.

Component Interfaces			
Interface ID	**Involved Components**	**Components Relation**	**Interface Content**
S-API.I.01	S-API and Third Parties	The S-API allows third-parties to authenticate themselves with the SPHINX system.	Third-party credentials. Authentication results.
S-API.I.02	S-API and SM	The S-API receives a list of available SPHINX services for third-parties, including the associated third-party interface specification from the SM.	List of security services available for third-parties, including the third-parties' interface specifications.

Table 2: SPHINX S-API Interface Specifications

2.4.2 SPHINX Interfaces for Third Parties

Through a generic interface, third-parties are able to access specific cybersecurity services: Data Traffic Monitoring (DTM), Anomaly Detection (AD), Security Information and Event Management (SIEM), Forensic Data Collection Engine (FDCE), Homomorphic Encryption (HE), Anonymisation and Privacy (AP), Sandbox (SB) and Blockchain-based Threat Registry (BBTR) services.

Component Interfaces			
Interface ID	**Involved Components**	**Components Relation**	**Interface Content**
S-API.I.03	S-API and Third Parties	The S-API allows third-parties to discover and retrieve services related with the SPHINX certification process.	List of services available to perform the SPHINX certification process.
S-API.I.04	S-API and SPHINX components with third-party APIs (DTM, AD, SIEM, FDCE, HE, AP, SB, BBTR)	The S-API allows third-parties to access the SPHINX components' third-party interfaces.	Third-party request for a specific SPHINX service. Services accessible in the SPHINX Platform.

Specific Interfaces (realise S-API.I.04):

Component Interfaces			
Interface ID	**Involved Components**	**Components Relation**	**Interface Content**
DTM.API.01	DTM and Third Parties	The DTM enables third parties to receive information regarding detected anomalous or suspicious data traffic.	Third-party request for the SPHINX DTM service. Abnormal and suspicious traffic data.
DTM.API.02	DTM and Third Parties	The DTM enables third parties to receive statistical information on collected data traffic (e.g., number of connected devices and connected users, data access type, bandwidth used per device and per user).	Third-party request for the SPHINX DTM service. Statistical information on collected traffic data.
AD.API.01	AD and Third Parties	The AD enables third parties to receive information regarding detected anomalies in system and user behaviour that constitute a threat.	Third-party request for the SPHINX AD service. Detected anomalies in system and user behaviour.
SIEM.API.01	SIEM and Third Parties	The SIEM enables third parties to receive information regarding incident-related information.	Third-party request for the SPHINX SIEM service. Log entries of security incidents and threats.
SIEM.API.02	SIEM and Third Parties	The SIEM enables third parties to receive information regarding security information and events.	Third-party request for the SPHINX SIEM service. Log entries of security information and events.
FDCE.API.01	FDCE and Third Parties	The FDCE enables third parties to receive information regarding new threats, following the detection of successful attacks.	Third-party request for the SPHINX FDCE service. New threat information (attack type information and metadata).
HE.API.01	HE and Third Parties	The HE enables third parties to encrypt sensitive information.	Third-party request for the SPHINX HE service. Encrypted sensitive data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

HE.API.02	HE and Third Parties	The HE enables third parties to perform searches on repositories containing sensitive data.	Third-party request for the SPHINX HE service (search query). List of files (matching query).
AP.API.01	AP and Third Parties	The AP enables third parties to anonymise personal data.	Third-party request for the SPHINX AP service. Anonymised personal data.
AP.API.02	AP and Third Parties	The AP enables third parties to anonymise personal data in traffic information.	Third-party request for the SPHINX AP service. Anonymised personal data in traffic information.
SB.API.01	SB and Third Parties	The SB enables third parties to receive information regarding the available SPHINX cyber security certification services for third-party components.	Third-party request for the SPHINX SB service (search query). List of available SPHINX cyber security certification services.
SB.API.02	SB and Third Parties	The SB enables third parties to receive information regarding their components' cyber security certification.	Third-party request for the SPHINX SB service (third-party component's technical specifications). Certification report (CVSS format).
BBTR.API.01	BBTR and Third Parties	The BBTR enables third parties to receive information regarding new threats and attack types.	Third-party request for the SPHINX BBTR service. New attack type information and metadata.
BBTR.API.02	BBTR and Third Parties	The BBTR enables third parties to receive information regarding the registration of new threats.	Third-party request for the SPHINX BBTR service. New attack type information and metadata.
BBTR.API.03	BBTR and Third Parties	The BBTR enables third parties to receive information regarding registered threats.	Third-party request for the SPHINX BBTR service (threat select criteria). List of registered threats meeting the provided criteria.

Table 3: SPHINX Third-parties Interfaces Specifications

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

3 Conclusion

This document presents the detailed design for the SPHINX Application Programming Interface (API) for Third Parties or S-API component, as introduced in the SPHINX architecture (D2.6) [1]. It delivers a general overview of the S-API, with the identification of its positioning within the SPHINX Platform, followed by a presentation of the S-API requirements that extended the technical specifications defined in [1] with the specific requirements for the component addressing its specificities. A set of sequence diagrams is also presented, illustrating key functions of S-API from the perspective of third-parties and SPHINX Services, allowing to better understand details pertaining to the implementation phase.

This document provides an important basis for the technical implementation of the S-API component in SPHINX.

The S-API component will be revised and updated as part of Task 3.6 - Third Parties Enabling APIs, considering the contributions from other relevant project outcomes, including the development work of the S-API component, the final version of the SPHINX architecture and the ongoing integration work in SPHINX.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4 References

[1] SPHINX Project. *D2.6 SPHINX Architecture v2*. Version 1.0. Dated 24-02-2020.

[2] SPHINX Project. *D2.4 - Use Cases Definition and Requirements Document v1*. Version 1.0. Dated 31/12/2019

[3] SPHINX Project. *D2.5 - SPHINX Requirements and Guidelines v1*. Version 1.0. Dated 31/12/2019

[4] IETF. *RFC 8259 - The JavaScript Object Notation* (JSON). December 2017. Available at: https://tools.ietf.org/html/rfc8259

[5] OpenAPI Specification. Version 3.0.3. Available at: http://spec.openapis.org/oas/v3.0.3

[6] IETF. RFC 8252 - OAuth 2.0 for Native Apps. October 2017. Available at: https://tools.ietf.org/html/rfc8252

[7] IETF. RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage. October 2012. Available at: https://tools.ietf.org/html/rfc6750

[8] SPHINX GitLab Available at: https://sphinx-repo.intracom-telecom.com

[9] Swagger. Available at: https://swagger.io/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.