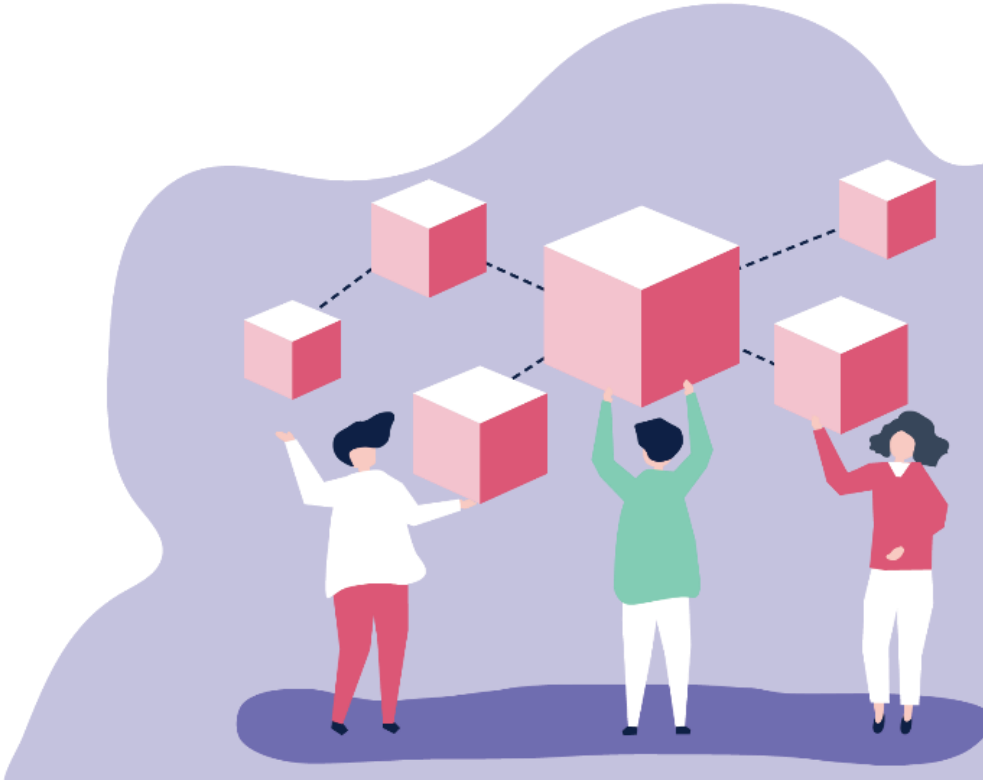
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 857223

D4.5 Gatekeeper Trust Authority

Deliverable No.	D4.5	**Due Date**	30/09/2020
Description	GTA implementation and integration		
Type	Other	**Dissemination Level**	PU
Work Package No.	WP4	**Work Package Title**	GATEKEEPER Things Management Infrastructure & Development
Version	1.0	**Status**	Final



Authors

Name and surname	Partner name	e-mail
Konstantinos Votis	CERTH	kvotis@iti.gr
Eleftheria Polychronidou	CERTH	epolyc@iti.gr
Daniel Rodriguez	Sense4Care SL	daniel.rodriguez@sense4care.com
Eugenio Gaeta	UPM	eugenio.gaeta@lst.tfo.upm.es
Leire Bastida Merino	Tecnalia	Leire.Bastida@tecnalia.com

History

Date	Version	Change
01/09/2020	0.1	Initial draft
25/09/2020	0.2	Added GTA Internal and External architecture and internal components conceptual and technical description
28/09/2020	0.7	Version ready for peer review
02/10/2020	0.8	Review from Sense4Care, UPM
02/10/2020	0.9	Review by coordination team
04/10/2020	1.0	Final version

Key data

Keywords	Trust Authority, Blockchain, Certification, Validation, Auditing, Trusted Data Sharing
Lead Editor	Konstantinos Votis
Internal Reviewer(s)	Daniel Rodriguez (Sense4Care SL), Eugenio Gaeta (UPM)

Abstract

This deliverable reports on the progress of T4.5 which refers to the design and implementation of the GATEKEEPER Trust Authority component that is mainly responsible

for providing Things Validation and Certification ensuring that the Gatekeeper Platform is private and secure by its design.

In this version of the deliverable we describe the initial version of the GATEKEEPER Trust Authority which is part of the initial version of the Gatekeeper platform along with the Thing Management System (T4.2) that is used as an entry point for the API infrastructure, and the Data Federation Framework (T4.4).

This deliverable will be updated in a second version due in M24.

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of contents

Abbreviations

Table 1: List of abbreviations

ABAC	Attribute Based Access Control
API	Application Programming Interface
CA	Certificate Authority
GTA	Gatekeeper Trust Authority
HTTP	HyperText Transfer Protocol
IDS	Intrusion Detection System
IDSA	International Data Spaces Association
NCP	National Contact Point
OWASP	Open Web Application Security Project
PKI	Public Key Infrastructure
SAML	Security Assertion Markup Language
SDK	Software Development Kit
SSO	Single Sign On
TMS	Thing Management System
WoT	Web of Things

List of tables

List of figures

1 Introduction

This deliverable describes the Gatekeeper Trust Authority (GTA) which is one of the core components of the Gatekeeper Platform as outlined in the Gatekeeper Logical Architecture described in Deliverable D3.2 [1] This component is based on prior CERTH experience in building the blockchain based logging/auditing and permission handling mechanism designed and developed in frames of the KONFIDO project [2].

This mechanism provides immutable logging of the audit messages exchanged between two National Contact Points (NCPs). These audit messages correspond to services called between different NCPs. Furthermore, it provides data sharing and data permission handling based on identities.

The GTA extends the functionalities provided by the logging and permission handling mechanism developed in KONFIDO by employing a validation mechanism that will validate Things in accordance with the Gatekeeper Thing profiles and the guidelines and standardisations provided in frames of T8.1. It also extends the basic Identity management service used by the KONFIDO mechanism by using (pluggable) Certificate Authorities (CAs) as the root of trust for managing the Certificates of the Things and by matching Thing Description (TD) properties as attributes of this Certificate.

GTA interacts with the Thing Management System (TMS) so as to provide for Gatekeeper Things Authentication and Authorisation. Furthermore, it leverages the extended Identity Management service mentioned above so as to allow for trusted Thing sharing among the Users of the Gatekeeper Platform.

The structure of this deliverable is as follows. In Chapter 2, we describe the concepts behind and the main functionalities of the GTA along with its position in the overall Gatekeeper architecture, the components of its inner architecture, and the specific security and privacy features that the GTA contributes to the Gatekeeper Platform. In Chapter 3, we describe the basic functionalities of its components and the technologies we used for their development. Finally, in Chapter 4, we describe how the GTA components are deployed in a development and in the production environment by outlining the integration, (i) among the inner components, and (ii) between the GTA and the TMS.

2 Gatekeeper Trust Authority Version 1

2.1 Position of the GTA into Gatekeeper architecture

The Gatekeeper Trust Authority (GTA) is a core component of the Gatekeeper Platform as described in the Logical Architecture of Gatekeeper in Deliverable 3.2 [1]. The position of GTA in the Gatekeeper Platform appears in Figure 1 below.

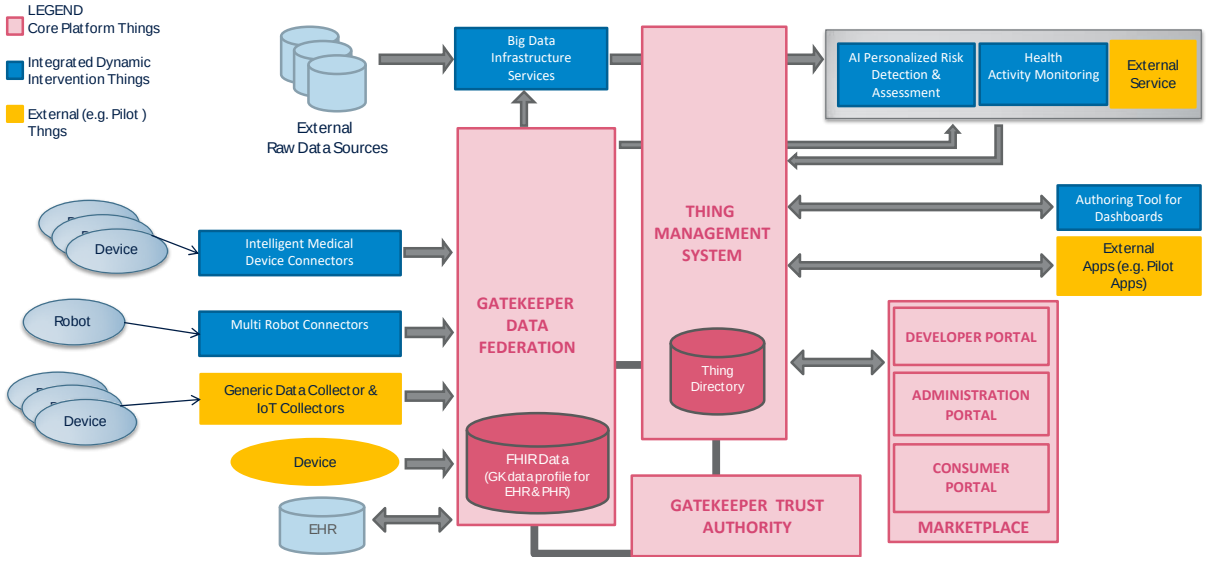


Figure 1 - Gatekeeper Platform logical architecture

The GTA is the component responsible for (i) the validation of Things based on a set of predefined standards and specifications, (ii) the certification of Things and the linkage of the Thing Certificate with the results of the Thing validation, (iii) the action tracking that allows for transparency and auditing, and (iv) the trusted Thing Sharing among the Users of the Platform following the FAIR principles [3].

GTA interacts with the TMS to authenticate Platform Users and authorise them to access the Things stored in the TMS Thing Directory subcomponent based on a set of predefined rules. Moreover, it interacts with the TMS

- each time an action (e.g. register, consume) is done on a Thing by a User
- each time a User performs an action on the Platform (e.g. upon the User authentication)

so as to log the actions in the GTA.

Moreover, GTA interacts with other core components of the Gatekeeper platform taking security and privacy by design features like security, privacy, confidentiality, data integrity, and stakeholders' non-repudiation into account. To do so, there are several technical means employed during the GTA integration with other Gatekeeper components such as HTTPs secured connections, IDSs, firewalls, and blockchain technology.

2.2 GTA internal architecture

In **Error! Reference source not found.**Figure 2 below, we present the internal conceptual architecture of the GTA and the foreseen interaction with the TMS.

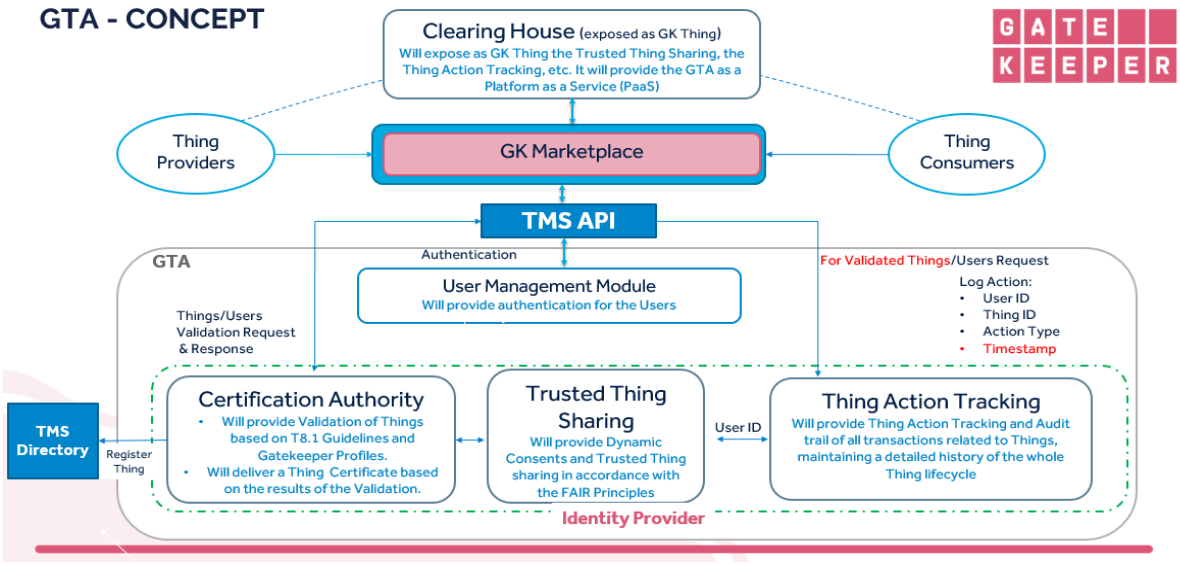


Figure 2 – GTA conceptual architecture

2.2.1 User Management Module

The User Management Module is provided as a microservice accessible to the other components of the Platform. It acts as an 'Authentication Server' used by all the components of the Gatekeeper Platform that need to provide User authentication. More specifically, whenever a User wants to authenticate with a component, the User is redirected to the User Management Module where the credentials of the User are verified. Upon successful credential verification, the User is authenticated to connect with other components of the Platform without a need to re-authenticate themselves. Moreover, it connects with the Thing Action Tracking to immutably log the actions of the Users and with the Certification Authority to provide the authorization rules for the Users as regards their access to Things.

2.2.2 Certification Authority

The Certification Authority is provided as a microservice accessible to the other components of the Platform. This component is responsible for the Things Validation and Certification. It validates the Things based on a predefined set of standards in accordance with the Gatekeeper Profiles and calculates a validation score based on the results of the validation. Then, it issues a Certificate for the Thing and registers the validated Thing to the Thing Directory by linking the Think description with the calculated Validation score. The Validation score is not regarded as sensitive information and as such, it can be transferred from the GTA to the TMS. Within the TMS, the Validation score is mapped with a Role which corresponds to a set of access control rules. It's worth noting that Users are also regarded as Things and, as such, they follow the same Validation process. Further details on the Validation process will be given in the next version of this deliverable.

2.2.3 Dynamic Attribute Provisioning (Trusted Thing Sharing)

This component leverages the Identity Management mechanism in order to allow Users to share their Things in a trusted manner by enabling dynamic consent The Things Sharing

follows the FAIR principles. The trusted Thing Sharing among the Users of the Platform following the FAIR principles [3] and aligned with the Dynamic Attribute Provisioning module of the International Data Spaces Association (IDSA) reference architecture [5]. Further details on this will be given in the next version of this delivery.

2.2.4 Thing Action Tracking

This component keeps an audit trail of all the actions done on Things during the whole Thing lifecycle. In particular, it logs the Actions performed by Users[1] such as the User logging to the Platform (User authentication), the Certification of a Thing by the Certification Authority, and the Consumption of a validated Thing by an authenticated User. The actions are logged in an immutable way and serve as a digital evidence enabling for transparency and auditability.

2.3 Component interaction and integrations

The GTA has a direct interaction with the TMS while it can also be used for providing authentication to any core Gatekeeper component.

2.3.1 Interaction and integration with the TMS

GTA is integrated with TMS for the User authentication and authorisation, for the Thing Validation and for the Thing Action Tracking. The interactions between the GTA and the TMS are the following.

- User Management module. The GTA exposes a set of APIs which can be consumed by the TMS in order to provide authentication for the Users. The details of the integration between the TMS and the User Management module are presented in Section 3.1 below.
- Certification Authority. Upon a new Thing Registration, the GTA validates the Thing descriptions of the Things and registers the Things in the Thing Directory component of the TMS by linking the result of the Validation, i.e. the Validation score with the Thing Description. The details of the integration between the TMS and the Certification Authority will be in the next version of this deliverable.
- Thing Action Tracking. When a User is authenticated, when a new Thing is registered, and when a validated Thing is consumed by an authenticated and authorised User, the actions are logged in the GTA. However, only is the latter case there is a direct interaction between the GTA and the TMS. The GTA exposes an API whose services are consumed by the TMS Gateway to log the action performed to the Thing. More specifically, when a User requests to access an endpoint of a Thing from the Thing Directory, the data corresponding to the action, are posted to the Thing Action Tracking. An example client request containing the data that need to be posted appears in Figure 3 below. A documentation of this API has been outlined in the GTA component description, in the Deliverable D3.2 [1].

[1] Users are regarded as Things in the Gatekeeper ecosystem

```
{"user": {
    "name": "foo",
    "email": "foo@org.gr",
    "ip": "127.0.0.1",
    "thing_id": "FHIR-server",
    "action": "consume",
    "timestamp": "1600342917"
}
}
```

Figure 3 – Example request to the Thing Action Tracking API

2.3.2 Interaction and integration with other components

The User Management Module of GTA acts as an 'Authentication Server' and has the potential to be used by all the components of the Gatekeeper Platform that need to provide a User authentication process.

2.4 Security and Privacy Features of the GTA

2.4.1 Features provided to the Gatekeeper Platform

The design goals of the GTA are aligned with the core purposes of the International Data Space Association [6] which aims to provide controlled exchange of any type of data between organisations. Blockchain technology, adopting a consortium model with well-defined governance, will be leveraged for Identity management and Attribute Based Access Control (ABAC) allowing for privacy and confidentiality of all the messages exchanged, while all the actions done on Things will be logged in a decentralised and immutable way allowing for auditing, accountability, and non-repudiation. Finally, the history of these transactions will be visible to the interested stakeholders of Gatekeeper allowing for data transparency.

2.4.2 Privacy and Security by design and OWASP compliance

The security and privacy features provided by the GTA aim to follow the security and privacy by design principles of the Open Web Application Security Project (OWASP) [7]. These principles aim to improve the security of software with focus on web applications. OWASP has published an awareness document describing the top 10 web application security risks [8].

In the table below, there is a mapping between the processes offered by the GTA and the corresponding OWASP tot 10 security concerns for web applications they aim to address.

Table 2 : GTA processes and OWASP security risks mapping

GTA process	GTA subcomponet	OWASP web application security risks

Authentication	User Management Module	Broken Authentication
Authorisation / Access Control	Certification Authority	Broken Access Control
Things Validation	Certification Authority	Using components with known vulnerabilities
Auditing / Logging / Digital evidence	Thing Action Tracking	Insufficient Logging and Monitoring

3 GTA Components implementation

The Gatekeeper Trust Authority is composed of four main components: the User Management Module, Certification Authority, the Dynamic Attribute Provisioning, and the Thing Action Tracking.

3.1 User Management Module

The User Management module acts as a common 'Authentication Server' for all the Gatekeeper Platform components that need a User Authentication process. It is as a Single Sign On (SSO) server that acts like a firewall on top of the Gatekeeper components.

The User Management module is implemented with keycloak [4]. Keycloak is based on standards and provided support for OpenID Connect, OAuth 2.0, and SAML.

All Gatekeeper components that need to authenticate Users redirect them to keycloak for authentication. Users login to keycloak and their credentials are validated against the credentials stored in a database connected with keycloak which is implemented with PostgreSQL 13. Upon successful validation, keycloak creates an authenticated session with the User and redirects the User back to component they came (i.e. back to the referrer url). The session contains a session id and a token.

The database has a table for user management with two fields per user, i.e. username and password. The password is one way hashed for security reasons. When the User logs in to keycloak, keycloak validates the username and the hashed password against the values stored in the database. Upon successful validation, keycloak creates an authenticated session which contains a session id and a token that contains encoded User information. The token is returned back to the caller component and the APIs of the components that have access to this token can use this token to authenticate with the keycloak.

3.2 Certification Authority

The Certification Authority provides the Thing Validation and Certification, using a Hyperledger Fabric Blockchain network [9] as its core technology. The peers of this network are managed by a defined set of organisations, called consortium. Only the consortium peers can participate in the validation of the transactions and manage the shared ledger in accordance with a set of predefined policies. The involved Gatekeeper stakeholders can leverage this consortium Blockchain model so as to have control over the validation and certification of the Things.

All the participants of the network have known identities. Hyperledger Fabric uses a Public key infrastructure (PKI) to identify the network participants and verify the actions they are doing on the network. Each node participating on the network and each client that submits transactions to the network need to have a public certificate and a private key so that they can prove their identities. All certificates are issued by an organization that is member of the network allowing for a root of trust. These organisations run their own Certificates Authorities.

Channels are layers of communication between specific network members and applications (clients) having verifiable identities belonging to the organizations of the network can submit transactions to these channels.

The business logic of the Things validation is embedded within chaincode (or else, smart contract) which is packaged code deployed on a channel. Applications with verified identities can invoke these smart contracts so as to validate new Things and obtain the Certificates for them, and query these smart contracts to check e.g. for valid Things, the validation status of a Thing, etc. These applications use the Hyperledger Fabric SDK for Node.js [11] and the 'fabric network' package [12] in order for them to interact with the network, and the 'fabric-ca-client' to interact with the Certificate Authorities servers to manage their Identities. The smart contract code, written in JavaScript in our case, uses the Node.js contract API for supporting the development of smart contracts along with its 'fabric-contract-api' package [13].

Access control is also made at the smart contract level by using specific attributes of the Identities of the applications that invoke smart contract. These attributes are encoded within the Certificates of these applications. This type of access control is called Attribute-Based Access Control, or ABAC [10].

3.3 Dynamic Attribute Provisioning

This component provides the mechanism for Trusted Thing Sharing and dynamic consent. It is based on the Hyperledger Fabric PKI described in Section 3.2 above for proving and verifying the Identities of the entities that share Things while the business logic of the sharing is encoded within smart contracts (chaincode) deployed on the Hyperledger Fabric network.

3.4 Thing Action Tracking

This component keeps an audit trail of all the actions (e.g. register, consume) done on Things by (authenticated and authorised) Users during the whole Thing lifecycle. The Identities of the Users and the Things participating on the action are proved and verified based on the Hyperledger Fabric PKI described in Section 3.2 above. The business logic for storing the audit logs in the ledger and for querying the audit logs is encoded within smart contracts (chaincode) deployed on the Hyperledger Fabric network.

This component uses a 'middleware API' deployed on a nodes.js server for integrating with the TMS gateway. The TMS gateway posts the details of the action done on a Thing by calling a dedicated service deployed in this middleware API. The node.js server uses the 'express', the 'body-parser', and the 'cors' packages for managing the API. An example client request appears in Figure 3 above.

3.5 Clearing House

The Clearing House provides the possibility to expose as Thing some of the core Gatekeeper subcomponents of the GTA, e.g. the Dynamic Attribute Provisioning (Trusted Thing Sharing) and the Thing Action Tracking. Further details on this will be provided in the next version of this deliverable.

4 GTA Deployment environments

The cluster's deployment framework is Kubernetes, allowing for scaling and maintainance the GTA without interfering with other services that may be inside the same cluster.

4.1 Development environment

The development environment is located at CERTH premises within an Oracle VirtualBox VM running Ubuntu 18.04. The development environment contains all the implemented components as dockerised services that reside at the same network and are configured and managed via docker compose. This environment is portable to the Kubernetes cluster that will constitute the production environment. The docker images of the running containers in the development environment for the main components appear in Figure 4 below. The containers correspond to the following components, two peers with one CA each, one orderer[2] with its CA, and two chaincode images deployed in the two peers of the network.

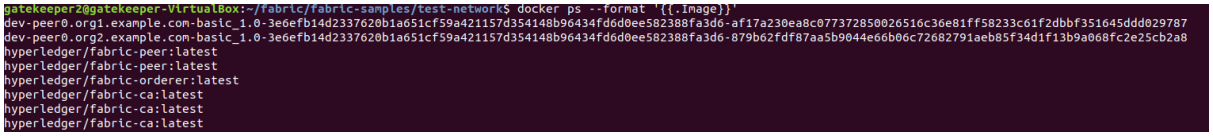


Figure 4 - The docker images of the running containers in the development environment

The development environment consists of a middleware API that contains the services that handle the requests from clients, then formulate and submit the transactions to the chaincode images that are deployed to the peers of the network and run as docker containers.
The middleware API contains the following services exposed to clients.

Table 3: Services exposed to clients contained in the middleware API

Service	endpoint	description
Create Action	http//<API-SERVER>:<API-PORT>/api/log	Logs to the blockchain the details of a new Action done by a User on a Thing
GetAllActions	http//<API-SERVER>:<API-PORT>/api/getAllActions	Returns the details of all Actions that are that are logged in the blockchain

[2] The orderer orders the transactions, creates new blocks of ordered transactions and sends the blocks to the peers of a channel to validate its transactions and commit the block to the ledger

GetAllUsers	http//<API-SERVER>:<API-PORT>/api/getAllUsers	Returns the details of all Users that have logged actions in the blockchain
Read Action	http//<API-SERVER>:<API-PORT>/api/readAction	Returns the details of an action given the action id
Read User	http//<API-SERVER>:<API-PORT>/api/readUser	Returns the details of an action given the user id
Action exists	http//<API-SERVER>:<API-PORT>/api/actionExists	Returns true if an action given an id exists, false otherwise
User exists	http//<API-SERVER>:<API-PORT>/api/userExists	Returns true if a user given an id exists, false otherwise

An example post request using a client that posts the data of a new action from a User appears in Figure XX below

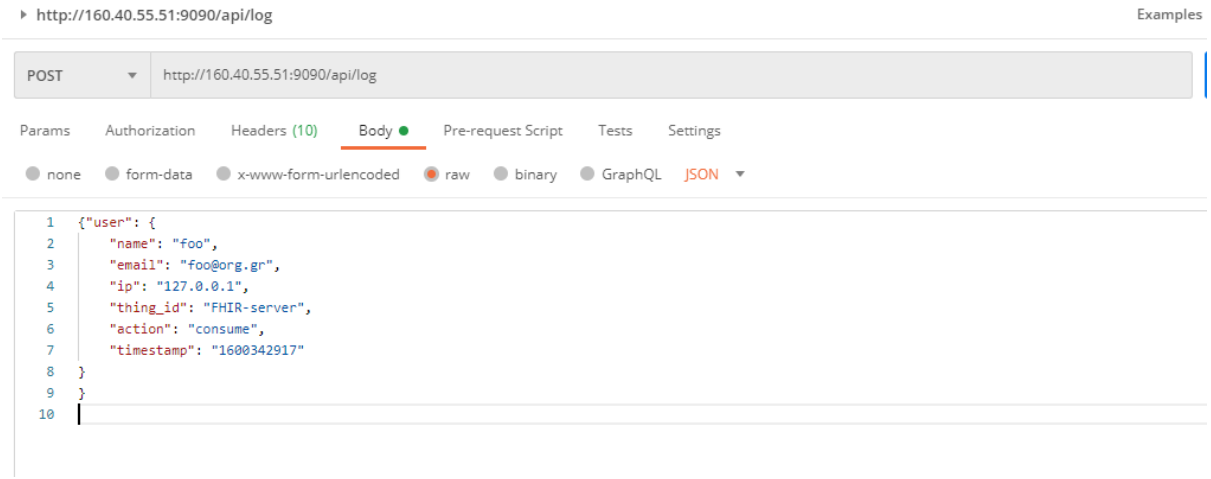


Figure 5 - Example client request for posting the Action data ta to the Blockchain

The details of the action and the user as logged in the Blockchain appear in Figure XX below.

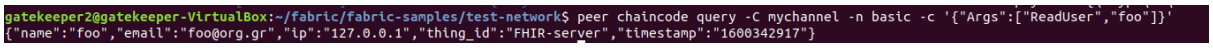


Figure 6 - Details of User and Action logged in the Blockchain

4.2 Production environment

The dockerised services will be deployed to the Kubernetes production cluster after appropriate configuration. Before proceeding to this step, the following issues will be taken into account.

- We will define the consortium by matching the organisations of Fabric with gatekeeper stakeholders. As such, the topology of the network will be properly

adjusted, e.g. the containers for the peers may run at different VMs located at the organisations' premises.

- Depending on the consortium formulation, we will examine the possibility of using multiple ordering nodes operated by one or multiple orderer organizations instead of a single node ordering service used by the development environment.
- We will examine the usage of CAs managed by the Organisations of the consortium instead of using the default CAs provided by Fabric.
- We will examine the full alignment of the GTA components with the IDSA components, e.g. the connectors, the proposed CAs, etc.
- We will examine the Idemix Credentials [13] and their privacy preserving authentication features as opposed to the Fabric CA Certificates.
- Certificate Revocation strategies will be defined in accordance with the consortium necessities.

5 Conclusions

This deliverable provides the initial description of the GATEKEEPER Trust Authority as well the position of the GATEKEEPER Trust Authority in the overall GATEKEEPER architecture and its integration with the Thing Management System which is the entry point of the API infrastructure. A development environment for the deployment of this component has been described. Due to COVID-19 situation, the demonstration in the production environment will be provided in the next version of this deliverable. The next version of this deliverable is planned for M24 and it will provide more detailed information on the methodologies and technical implementation of the Thing Validation and Certification, and of the Trusted Data Sharing, as well as technical details on the deployment of this component to the Kubernetes cluster.

6 References

[1] GATEKEEPER Consortium, Deliverable D3.2 Overall GATEKEEPER architecture
[2] https://konfido-project.eu/
[3] https://www.go-fair.org/fair-principles/
[4] https://www.keycloak.org/
[5] https://www.internationaldataspaces.org/wp-content/uploads/2019/09/Blockchain-Technology-in-IDS.pdf
[6] https://www.internationaldataspaces.org/wp-content/uploads/2019/03/IDS-Reference-Architecture-Model-3.0.pdf
[7] https://owasp.org/
[8] https://owasp.org/www-project-top-ten/
[9] https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html
[10] https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/users-guide.html#fabric-ca-client
[11] https://hyperledger.github.io/fabric-sdk-node/
[12] https://www.npmjs.com/package/fabric-network
[13] https://hyperledger.github.io/fabric-chaincode-node/
[14] https://hyperledger-fabric.readthedocs.io/en/release-2.2/idemix.html