A CIP-PSP funded pilot action
Grant agreement n°325188

Deliverable	D1.2.1 Specification of Tool Group “Centralised Data Clearing House”
Work package	WP1 Requirements & Specifications
Due date	M12
Submission date	31.01.2014 (Re-Submission 03.09.2014)
Revision	2.5
Status of revision	FINAL
Responsible partner	ECO
Contributors	Michael Weirich (ECO) Thorsten Kraft (ECO) Peter Meyer (ECO)
Project Number	CIP-ICT PSP-2012-6 / 325188
Project Acronym	ACDC
Project Title	Advanced Cyber Defence Centre
Start Date of Project	01/02/2013

Dissemination Level	
PU: Public	X
PP: Restricted to other programme participants (including the Commission)	
RE: Restricted to a group specified by the consortium (including the Commission)	
CO: Confidential, only for members of the consortium (including the Commission)	

Rev.	Date	Author	Notes
V0.1	06/18/2013	Peter Meyer (ECO)	Initial structure of the document / Table of content
V0.2	06/25/2013	Peter Meyer (ECO)	Added first details
V1.0	30/01/2014	Michael Weirich (eco)	API Keys/ JSON container/ File in and output / Examples for access the CCH / proposed User management CCH <-> CP
V2.0	16/06/2014	Michael Weirich (eco)	Restructuring of Document
V2.1	18/06/2014	Peter Meyer (eco)	Privacy and Legal Topics
V2.2	10/07/2014	Michael Weirich (eco)	Tables and Figures
V2.3	23/07/2014	Michael Weirich (eco)	Data Sharing / Access Management
V2.4	15/08/2014	Peter Meyer (eco)	Data Formats
V2.5	03/09/2014	Peter Meyer / Thorsten Kraft	Final Review

Glossary

ACDC	*Advanced Cyber Defence Centre*
CCH	*Centralized (Data) Clearing House*
SME	*Small-and Medium Enterprises*
DoW	*Description of Work, Part B Form of the Amendment*
NSC	*National Support Centre*
ISP	*Internet Service Provider*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
IdMS	*Identity Management System*
TI	*Trusted Introducer*
STIX	*Structured Threat Information Expression*
API	*Application-Programming-Interface*
SotA	*State of the Art*
HTTP	*Hypertext Transmission Protocol*
TTL	*Time-to-Live*
WP	*Work Package*
DAM	*Data Access Management*
EU	*European Union*
CERT	*Computer Emergency Response Team*

Index

List of Figures

List of Tables

1. Introduction

This deliverable **D.1.2.1 Specification of the Tool Group Centralised Data Clearing House"** is the first iteration of the specifications for the databases and data storage components within the ACDC project. This document provides the basic specifications and requirements for this key component of the project and supplies the information, how to integrate and interact with other project software components, as described in the **D1.1.1 "Overall Software Architecture".** A second document (D.1.2.2) at the end of the project (M30) will describe the actual implementation and adaptions that have integrated during this project pilot.

This document focuses on initial concept and design, focussing on the data itself, its storage, data formats, access control, user management and security framework.

The Central Data Clearing House with the internal abbreviation **CCH** is the central data storage repository of the ACDC project. The overall concept of the ACDC-project, the ACDC-approach, requires the CCH to limit the boundaries that might prevent other partners or stakeholders from providing and retrieving data. Therefore, the CCH has been designed to provide open standards and a wide flexibility in supporting data formats. This approach is necessary, as the ACDC project is designed as an integration pilot, collecting and processing from already existing tools, sensors, sources and further unspecified components.

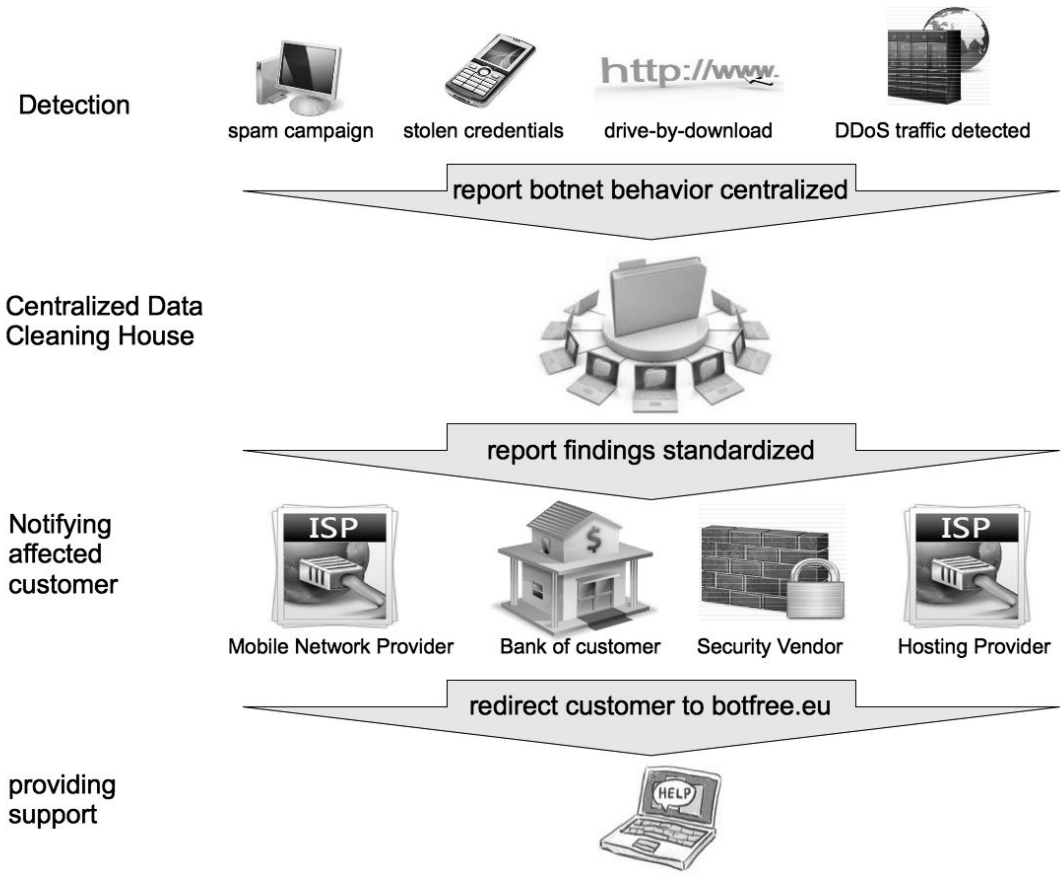


Figure 1: ACDC big picture

Even though the initial concept defined having no limitations on data (format) submissions, it has been determined across the project participants, that a basic standardisation of the submitted data fields and basic requirement on mandatory fields simplifies the data submission and retrieval. These specifications have been defined as the ACDC - "Schemata". These have been outlined and defined in the Deliverables D1.7.1/2 "Data Formats Specification".

Besides the data format specifications, same consensus had to be made for all other components of the overall software architecture. These have been identified within ***Work Package 1 "Requirements & Specifications"*** (WP1) of the ACDC project.

The following table displays other deliverables that specify the requirements of tools of the overall software architecture:

Deliverable	**Name**
D1.1.1 / D1.1.2	Overall Software Architecture Description
D1.2.1 / D1.2.2	Specification of Tool Group "Centralised Data Clearing House"
D1.3.1 / D1.3.2	Specification of Tool Group "Support Centre"
D1.4.1 / D1.4.2	Specification of Tool Group "Malicious or Vulnerable Websites"
D1.5.1 / D1.5.2	Specification of Tool Group "Network Traffic Sensors"
D1.6.1 / D1.6.2	Specification of Tool Group "End Customer Tools"
D1.7.1 / D1.7.2	Data Formats Specification
D1.8.1 / D1.8.2	Legal requirements

Figure 2: List of Deliverables WP1

Besides the collection and processing of data, the Centralized Data Clearing House is also designed to distribute data to stakeholders like ISP´s, Government agencies, Law Enforcement, Research groups or Industry Partners.

Due to security concerns, but also with the project aim to support an open community of stakeholders, the access management has been integrated into the Community Website / Community Portal of the project and the and user policy enforcement is part of the CCH.

One approach of ACDC is mutual data sharing across international boarders and an advanced capability of interaction, permissions and restrictions between the involved stakeholders. The Community portal manages the stakeholder / User database and maintains their relationships. This portal and further settings are described in the deliverable **D6.2.1 ACDC Social Platform.**

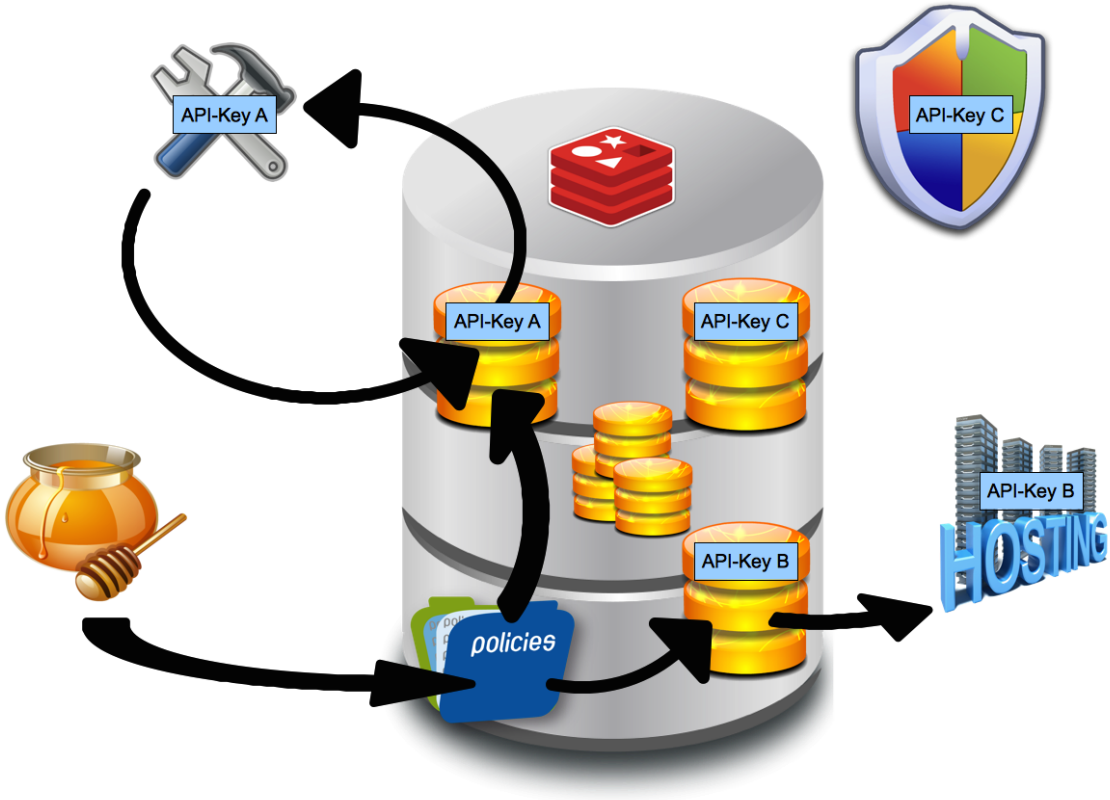


Figure 3: Policy based data exchange throuh the CCH

The community platform manages the access policies for the stakeholder and allowing each contributor full control about his submitted data. Restrictions could for instance deny data sharing with competitors or based on legal requirements. Access is granted by API’s, ensuring that the CCH cannot be accessed without the proper code that the community platform provided just for this particular request or for a given timeframe.

2. Privacy and Legal Requirements

The overall legal concept of ACDC has been addressed in the deliverable **D1.8.1 Legal Requirements**. Conclusions and recommendations from this and the general legal requirements have been incorporated into the development and conception of the CCH.

Generally, all ACDC partners shall follow the following guidelines:

Whenever personal data is processed, various legal bases must be complied with.

As the CCH as the central storage facility of the ACDC project collects and transfers IP addresses (static and dynamic) and domains to stakeholders of the entire project. The CCH has to consider the handling, storage and processing under legal requirements.

2.1. Dynamic IP addresses and personal data

Whether dynamic IP addresses can qualify, as personal data is disputable, because a clear assignment is not possible, unlike in the case of the static IP address. The starting point for the difference of opinion is the element of "determinability" *according Section 3 I Federal Data Protection Act.*[1]

The assignment of a dynamic IP address through the Internet access provider is merely temporary. In this way, the anonymity of the Internet user is guaranteed. Even if the IP addresses can be identified by the server operator, a long-term association between the address and a name, which would result in the user becoming known, is not possible. From the IP address as such there exists no direct relationship to a particular person, so that this first needs to be ascertained.

As the access provider in their own right undertakes the allocation of IP addresses, and because it is relatively uncomplicated for the access provider to ascertain this relationship between an IP address and the person, the above-mentioned cases where other persons, such as the mailbox provider, collect and forward dynamic IP addresses, remain disputable.

2.2. Relativity of the Relationship to the Person

One argument/legal opinion deals with the relativity of the personal reference, and applies according to Section 3 I Federal Data Protection Act for the assessment of the determinability, based on whether the responsible party can ascertain the relationship to a natural person with the means normally available to them and without disproportionate effort. In particular, a differentiation is made on the basis of whether the de-anonymisation is possible with proportionate effort. This should, however, only be possible for the access provider. A third party (here, the mailbox provider) is only able to identify the user behind the IP address with the help of the access provider, who, in turn, is legally not permitted to provide this information to third parties. The theoretical possibility of an identification of the user does not correspond to the aforementioned definition of determinability.

2.3. Objectivity of the Relationship to the Person

According to this argument/legal opinion, it is not relevant whether a disproportionate effort is required to de-anonymise the IP address. It is sufficient simply that the theoretical possibility of a link of any form exists between the email address and a natural person. This is regardless of that fact that the determinability of the person in the legal sense is only possible when the person is identified using legal means. In this argument, the Data Protection Act is specifically designed to protect against the misuse of data, meaning that such a limitation of the term determinability would not be justifiable. The objectivity of the relationship to the person is also supported by recital 26 of the *Data Protection Directive 95/46/EG.*[2]. The Art. 29 Working Party also assume the absolute definition of the term. In recital 26 of the EU Data Protection Directive 95/46/EG, it is unambiguously defined that all means that could be used by the party responsible for the processing, or by any other party who could be reasonably considered, are to be taken into account in order to establish whether a person is determinable.

[1] http://www.gesetze-im-internet.de/englisch_bdsg/englisch_bdsg.html
[2] http://www.dataprotection.ie/docs/EU-Directive-95-46-EC-The-Recitals-Page-1/90.htm

As it cannot be ruled out that third parties possess the additional knowledge required for the ascertainment of the relationship to the person, it depends in actuality on the judgment of the probability of a potential identification. Dynamic IP addresses can also be used by third parties, with the help of log files of the Internet Service Provider's (ISP) individual connections, leading potentially to the identification of the individual person. Therefore, it must be assumed that it is possible to ascertain the relationship to the person for dynamic IP addresses and that the data protection laws are applicable.

2.4. Legality of data processing in Germany

The physical hosting location of the CCH is in Germany, therefore, additional regulations and law considerations have to be taken into account. In General, personal data processing is legal in Germany, if the prior consent of the data subject is granted or of law permits it. The evaluation of national Law related to the objectives of the ACDC project refers to the following requirements.

2.4.1. Prior consent

According to Section 4 of Federal Data Protection Act the collection, processing and use of personal data shall be admissible only if permitted or prescribed by this Act or any other legal provision or if the data subject has consented. Further, the collector of personal data (the CCH) must comply with Section 4 a of Federal Data Protection, in particular the Data subjects shall be informed of the purpose of collection, processing or use and, in so far as the circumstances of the individual case dictate or upon request, of the consequences of withholding consent.

2.4.2. Permission by law

According to Section 28 (1) of Federal Data Protection Act the collection, storage, modification or transfer of personal data or their use as a means of fulfilling one's own business purposes shall be admissible when needed to create, carry out or terminate a legal obligation or quasi-legal obligation with the data subject, in so far as this is necessary to safeguard justified interests of the controller of the filing system and there is no reason to assume that the data subject has an overriding legitimate interest in his data being excluded from processing or use, if the data are generally accessible or the controller of the filing system would be entitled to publish them, unless the data subject's legitimate interest in his data being excluded from processing or use clearly outweighs the justified interest of the controller of the filing system.

2.4.3. Justified interests:

The data processing is necessary within the meaning of based on the collection, storage and transmission of strange data, so data subjects can be informed and further investigation can be initiated.

Whether the legitimate interests of the person concerned predominate, must be examined in the context of a balance of interests. Here the informational self-determination must be considered, which results from Article 1 und 2 of Basic Law for the Federal Republic of Germany (Grundgesetz, GG). An argument against an overriding legitimate interest of the person concerned is that only conspicuous data will be stored.

Moreover, it is precisely in the interest of the person concerned, when he gets informed about the conspicuousness regarding his data. The person concerned may take further steps for investigation and can take action to prevent further abuses its data. As far as the data processing serves the recognition and containment of data misuse and moreover, to avoid massive damage and major disruption to the telecommunications infrastructure, the collection and transmission of personal data is justified.

As a result, the data processing in accordance with the principles set out above is admissible. It is possible either to take action on the data subject's prior consent or a statutory basis.

2.5. Granularity of the shared data

The granularity of data that individual tools supply to the CCH will vary considerably. Some tools will be reporting finite lists of IP addresses that represent command and control servers for specific types of botnet, others will simply be reporting that TCP/IP connections have been received to IP addresses that have no services and are potentially suspect. A malware sample could be extremely detailed, including hashes of the file, the libraries it utilises, registry entries it creates, etc. It will therefore be

very important that the Data Exchange format used is able to represent data that varies in granularity. It is intended that the correlation capabilities of the CCH will allow data granularity to be increased when required by cross-referencing data fed by multiple tools to build a more complete picture of activity.

ACDC data controllers (CCH and Concentrators) shall put in place mechanisms to enable data subjects to enforce their right of access and related rights (right to confirmation, right of access to one's data, and right to rectification, erasure and blocking). The API key system to be used in the community platform, while designed as a data confidentiality and security mechanism, holds the potential to also be used as a tool to facilitate the exercise of data subject rights. Through the API key system, the platform has an overview of which data is sent to the CCH and to whom the data relates. Such control is likely to help the CCH respond to the exercise of data subject rights in a fair and responsible manner.

The Centralised Data Clearing House or CCH is the main element of the ACDC project. Via this platform, data collected through the detection tools will be aggregated and further analysed with the purpose of generating data feeds to interested controllers. This single EU common report will facilitate information exchange across the Union and enabling appropriate response against malware. The CCH is thus a gathering platform, which combines data collected by other group tools directly placed in public networks and personal devices to be further analysed. The findings of the CCH are thus shared with trusted partners that have previously requested to receive the data feed. To subscribe to the data feed, one must register via the community platform, which will grant access by API keys. However, access to the CCH data feed is limited in light of the specific and legitimate interest a party may hold. The relation between requests and nature of access will be measure and granted by the community platform based in the relationship and level of trust of a given party.

Before sharing any slots of data with the platform, countries are expressly requested to pay due attention to their national laws. This is to say that partners, as controllers of the processing that goes on in the network traffic sensors and malware website analysis tools, hold full responsibility with complying with national legislation before engaging in information sharing. Such duty must be borne by partners contributing with input data due to the impossibility of having the CCH verifying such compliance.

The CCH operates as a controller regarding the information received in the platform and bears responsibility for the processing thereof. The agent managing the CCH in casu ECO (Association of the German Internet Industry) is the controller of the platform. This control covers a three stage processing, which includes access to the data shared by data collectors (such as ISPs, end-users, computer security companies, banks, etc.), further processing relating to the analysis and aggregation of the data received, and a final processing concerning the distribution of this data with trusted parties according to the rules of the community platform. Once again, it is important to recall that the automation of the tools deployed in the CCH minimise human intervention to the maximum and information distribution in a need-to-know basis assures that only data pertaining to the network and interest of a partner is shared with such. This is also to say that only parties holding legitimate interest over personal data will receive redistribution of this data. It also means that not every partner receiving feeds from the CCH will have access to users' personal data, for the reasons explained above.

2.6. Data privacy considerations

With many types of shared data the sensitivity may depend on if the data is describing the victim of a cyber-incident or the perpetrator. Also objects such as files and e-mails could contain either sensitive or non-sensitive information. Categorising data to indicate the privacy issues may be required, along with the ability to anonymise information when it is shared. An illustration of the potential types of data objects that could be stored in the CCH as part of the ACDC project, and the potential privacy concerns can be found in Annex: *CybOX*[3] *Cyber Observables and Privacy*.

WP1 Deliverable 1.8.1 defines the legal issues regarding the dissemination of information, and technical solutions delivered by WP2 will be required to conform to the findings of this deliverable.

Article 7(f) requires a proportionality assessment, i.e. the legitimate interest of the data controller (ensuring the security of the network) must be weighed against the need to protect data subjects' fundamental rights (confidentiality of communications). In the case of ACDC, the need to fight botnets (as part of the security of the network) should be balanced against the need to protect users'

[3] https://cybox.mitre.org/

confidentiality of communications. While the need to ensure the network is protected against botnet is clearly legitimate, the proportionality assessment will bear upon the adequacy and necessity of the means used to achieve this goal, thus on the design of the ACDC solution. In this case, two data processing operations should be legitimized: the sharing of personal data with the CCH and its further processing for fighting botnets. This processing will be performed by the CCH and shared with a legitimate interested party, like ISPs, webmasters and hosts.

The necessity of the creation of a platform such as the CCH to efficiently fight botnets is justified by the international character of cybercrime and the significant threat posed by botnets. Providers, webmasters, hosting services, and computer security companies cannot tackle large-scale infections individually. The need for cooperation and information sharing is thus crucial. The automated and non-human operated character of the CCH reduces chances of unauthorized or abusive intervention.

With regard to the adequacy of the means used, the CCH environment consists of data processing that have a strong and concrete potential to reduce botnets across the EU. The CCH is thus a suitable and adequate environment to the objectives pursued: the tools deployed at the clearing house are tested and efficient mechanisms capable of promoting knowledge and sharing data of infections. This will facilitate disinfection, as the news feed provided by the CCH will enable partners to alert their infected users and redirect them to the National Support Centre for cleansing. The purpose of the CCH is thus protecting end users and networks from botnets, promoting prevention, detection and disinfection. The information processed at the CCH is not intended to be used against individuals, left alone the possibility of ISPs, webmasters and web hosts to take the necessary legal measure to safeguard their networks against malicious users. To this end, ISPs and all partners are requested to, in accordance with their national law, decide upon the remedies, sanctions and means to be given to users to enable their adequate defence. It is thus a duty of data controllers to ensure that data subjects' rights are fully respected and that the necessary remedies and opportunities for their exercise are put in place.

Finally, the rules of the community platform together with the numb and automated character of the CCH are designed to ensure the minimal possible impact on the fundamental rights of data subjects in being taken into account. Removing the processing of personal data would diminish the capability of the CCH to a level where the purpose is no longer attainable. However, input and output data are controlled and ranked by the community platform, which also enables sharing preferences and restricts access to data according to partners' legitimate interests. For instance, ISPs receiving data feeds from the CCH can only have access to information related to their own range of IP addresses, never to data relating to users that are not related to their services. This guarantees sufficient levels of confidentiality of communications and lesser invasive means on user data.

3. Data Storage

The CCH, same as the overall ACDC software architecture **(deliverable D1.1.1)** is designed to be flexible to limit the barriers that might limit stakeholders from submitting data. Particular to the CCH, this flexibility refers to the way to receive data from the different tools or aggregators. Though the CCH is able to receive data in a wide range of formats, certain mandatory and basic requirements have been defined and specified. These standard requirements are part of the standard ACDC Data exchange format. These data formats have been defined as part of the work in WP1, which is dedicated to standards and specifications. The deliverables outlining these requirements are **D1.7.1/2 "Data Formats Specification"**

The long-time data storage functionality of the CCH supports data correlation and basic data analysis, providing basic functionalities like:

- High data capacity with easy and seamless storage expansion;
- The ability to store data represented by the ACDC Data Exchange Format with minimal mapping to storage types;
- Support for fast querying of stored data;
- The ability to create triggers and watch lists in order to facilitate automated analysis;
- The ability to restrict views of information based on the permissions of the viewer.

3.1. CCH Infrastructure

All servers of the central infrastructure will physically stored and operated in a ISO-27001-certified Data-Centre. The data centre has an electronic access control system, with a 24/7 camera monitoring of entrances and server rooms secure the servers against unauthorized physical access. Remote access is only possible via SSH with Key Authentication and limited to a very few system administrators, with user rights and the possibility to get "root"-access via "sudo" only. The default root login is disabled by default.

The web servers as entry points for the entire system are completely independent systems to the database server. They run on the latest stable Debian version, access is only possible through SSH via key authentication. The CCH uses an NGINX, configured as a reverse proxy, listening on port 3000 (SSL on). All NGINX proxies connect to a "Thin Webserver" which is used as an API server and it is not reachable from the Internet itself, because it did not bind any TCP sockets by itself.

All other servers (Database and Filestore- Server) are in a virtual network. They are only accessible through the API-providing web server, with a valid API key, and for system administration purpose through SSH via key authentication only.

3.2. Database

The ACDC database consists of the following infrastructure:

- **Redis Database**[4]
- **Cassandra Database**[5]

The CCH feeds a Redis Database. The big advantage of Redis in this context is the ability to hold the whole dataset in memory. This in-memory nature of Redis has been chosen as it allows an extremely well performance compared to database systems writing every request or event into disk.

As the Redis database does not keep data for long time storage purposes, a Cassandra Database has been attached to the Redis Cluster. This Cassandra database stores and provides the data that is needed within the project, e.g. for long-time measurements, research or historical data.

Both databases, Redis and Cassandra, support clustering. This gives the system the possibility to increase performance and storage possibilities with no downtime or interruption to applications.

[4] http://redis.io
[5] http://cassandra.apache.org

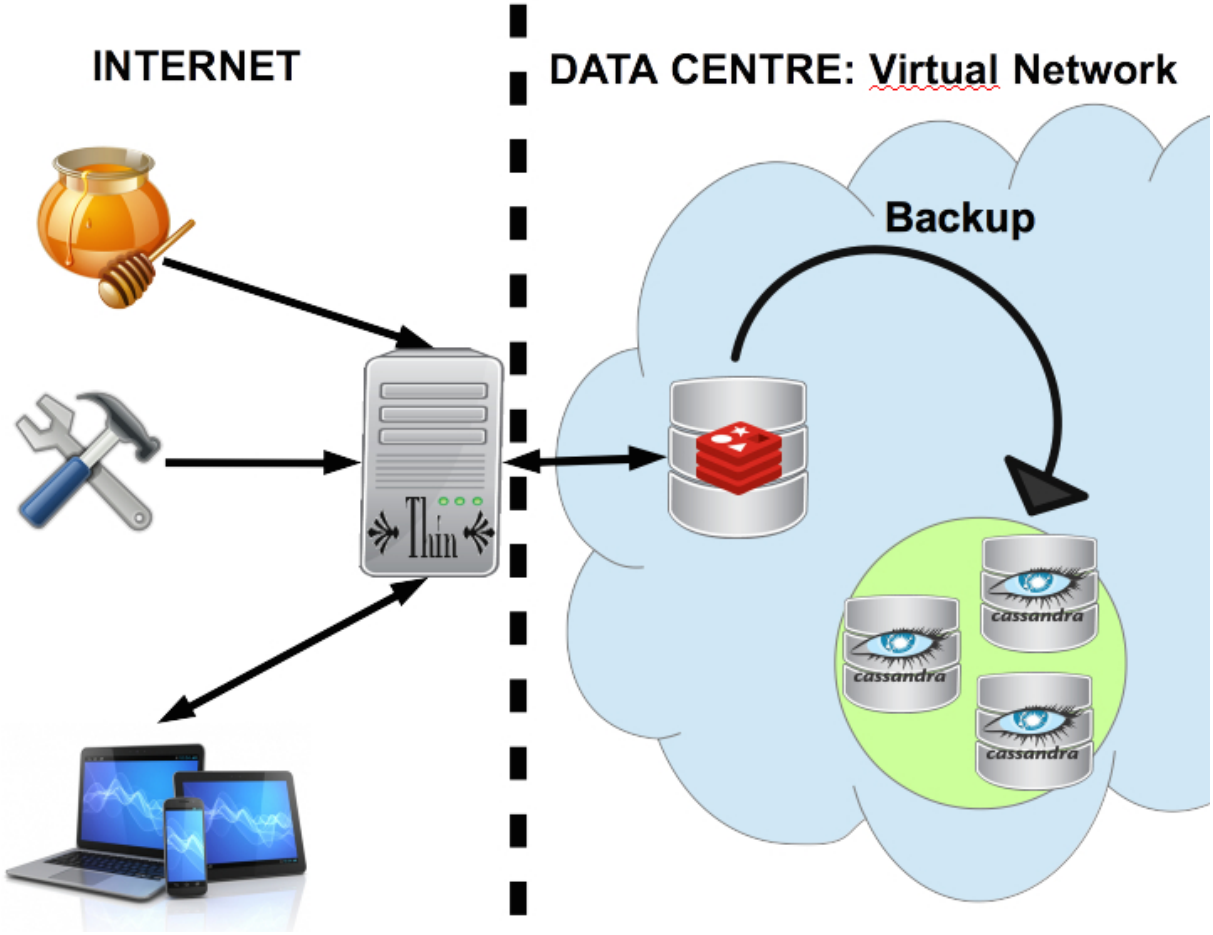


Figure 4: CCH Infrastructure

3.3. Tools, Sensors or Aggregators

Tools, Sensors or Aggregators can only access the CCH through unique API-keys. An API-Key of any external component needs to authenticate itself through the API-Server, which is located on both databases (Redis and Cassandra). The specifications for any external tools have been defined in the following deliverables:

- D1.4.1/2: Tools for Malicious or Vulnerable Websites analysis and detection
- D1.5.1/2: Network Traffic Sensors
- D1.6.1/2: End Customer Tools

External Sensors enrich the database with additional information. This data feed adds new incidents or enriches existing incidents with additional information on possible activities related to botnet.

A sensor can deliver incidents or details on a given incident to the CCH, which another sensor needs to perform in his task. In this case the second sensor should be hooked to a listener, watching the relevant data stream for his task.

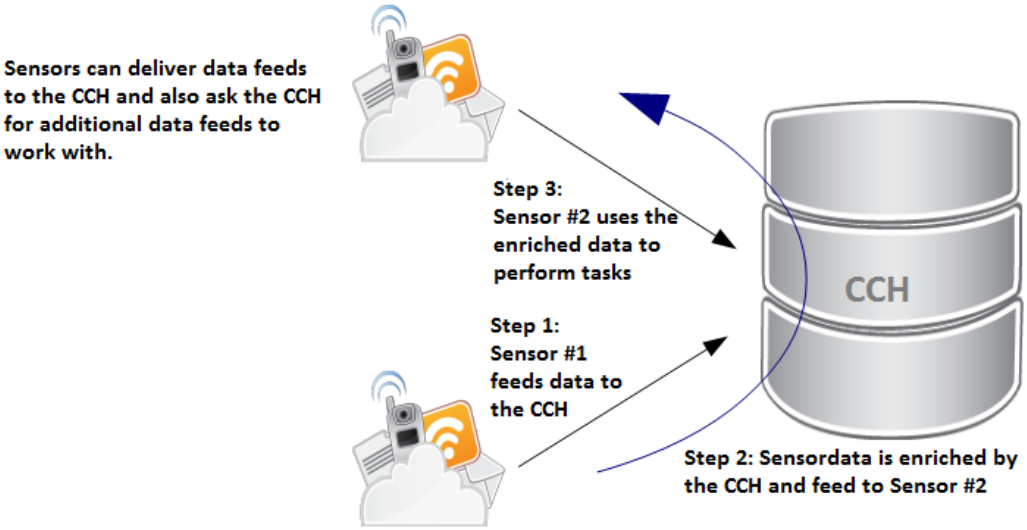


Figure 5: Sensors feeding data into the CCH

Each sensor has its own unique API-Key and the API-Key also includes the information about the schema for automated processing. Schemata are further explained in chapter 6.1 of this document and in deliverable **D1.7.2 Data formats.**

3.4. Data Aggregation

The CCH has a minimal logic of own data aggregation. This covers the functionality of increasing the counts for a certain event, the grouping of same-type incidents, the grouping by file-hash, domain, IP or ASN, the addition of a TTL or basic information like first-seen, last-seen.

4. User Rights Management

The Centralized Data Clearing House collects and stores data from different sources. This data, along with additional information enriched through ACDC intelligence, will be distributed to either the data provider itself, or to a third party with the privilege to receive such data.

Every data contributor will gain full control about his submitted data. This policy has been introduced, as either legal boundary restricts data sharing with stakeholders outside a certain country or region, others may have concerns sharing their data with Law Enforcement agencies, and other partners simply don't want to share their knowledge with the competitors, like Anti-Virus companies or rivalling Research Organisations.

The Philosophy of ACDC follows the directive that data sharing follows the mutual approach within a community. An organisation not willing to share data with their competitor will not receive access to that organisations' data either. Such policies are handled through the community portal, the front-end of the entire project. User need to apply through the ACDC Community Portal (https://communityportal.acdc-project.eu), and the permissions and stakeholder relationships are managed by the portal.

The portal itself allows a participating organisation to specify with whom they are going to share data and with whom not. The specifications can be set based on different stakeholder groups, regional settings or directly to a certain organisation. The model and concept, same as the different user groups, are handled in the deliverable **D6.2.1 ACDC Social Platform**, namely in the chapter 8, "The ACDC Portal Applications".

4.1. Security Specifications

The Community Portal maintains the access management of the CCH, and the various data sources are graded depending on their level of trust. Moreover, data feeds are provided on-demand and limited in light of national laws and special interests of stakeholders. To the extent that the CCH receives and releases non-personal data, no legal obstacles can be raised.

The Community Portal user right management has been designed as mechanisms to ensure the minimal possible impact on the fundamental rights of data subjects. Removing the processing of personal data from this context (e.g. IP addresses, domain names, email addresses) would diminish the capability of the CCH to a level where the purpose is no longer attainable. Input and output data are controlled and ranked by the Community Portal, which also enables sharing preferences and restricts access to data according to partners legitimate interests.

Therefore, ISPs receiving data feeds from the CCH can only have access to information related to their own range of IP addresses, never to data relating to users that are not related to their services. This guarantees sufficient levels of confidentiality of communications and lesser invasive means on user data. The only machine that is visible to outside and the only machine, which will or can communicate to outside the internal network, is located at *webserver.002.eco.redacted*.

The real-time access to the database is handled through so-called Channels. Each owner of an API-Key receives access to the real-time database, enabling immediate notifications about events he is legally allowed to receive only through his dedicated channel. The setting and registration of an ASN owner is handled through the Community Portal, additionally secured by an API-Key and authenticated through the planned implementation based on the policies like in the *Trusted Introducer* [6] approach.

[6] https://www.trusted-introducer.org/

4.2. Reputation & Trust

The operator of the CCH, namely ECO, currently determines the reputation and trust of data submitter manually. Their judgements are based on the extended experience in the IT-Security, following guidelines like

- Data quality of previous submissions, determined from other projects, initiatives and partnerships.
- Reputation of the data provider in the industry, 2nd opinions.
- Attribute of the submission (Unfiltered data from an end-customer tool like a report button to submit Spam)
- Pre-verification, e.g. in an aggregator
- Type of organisation (e.g. a AV-Company is supposed to submit verified data)
- Trust by others, 3rd party recommendations (e.g. at Virus Total)
- Known vs. unknown data source
- Frequency of a data feed (Single, continuous or periodic feed or data provider)
- Error rate, false positives
- etc.

In this stage of the project, no detailed processes for the determination of reputation and trust has been defined yet. The partners simply have to trust each other and their expertise in this realm, same as to trust all other partners to provide quality data. Following the community approach, each partner still has to consider using own quality assurance measurements, before adapting data into his own environment or products. In business, a Quality Assurance department would handle such responsibility, but as ACDC is a pilot project, the objectives just require a proof that the detection of botnets and malware are feasible with this current implementation.

Long-term plans, based on a sustainability and exploitation plan, will have the implementation of such QA measurements on the "product roadmap". The vision is to combine such intelligence with a self-learning and community-supported approach.

4.2.1. Trusted Introducer

The implementation of the Trusted Introducer[7] (TI) approach as a model for authentication has been added to the ACDC roadmap. This service established by the European CERT community was identified as a good approach how to safeguard a trusted environment within the ACDC-Community.

Following their concept, the TI-adaption within ACDC should ensure that community members especially involved in data sharing or decisions-and policy-making meet a defined level of trust, credibility, ethics or standards. In case of the participating CERTs within ACDC, a direct implementation of Trusted Introducer has been considered and set to a stage of evaluation.

[7] https://www.trusted-introducer.org/

4.3. API-User Types

The API concept is based on a hierarchic model of a three-stage API-Key distribution (CCH-Manager/CCH-Key-Manager/CCH-Key-User). Besides CCH-Manager-Keys, the community-portal is able to assign individual keys to subscribers of the ACDC community. With such a Key, each partner (CCH Key Manager) is capable of creating and assigning individual API-keys (CCH Key User) to own sensors, nodes or even sensors running on end-customer devices through the community portal, where each key gets registered.

Based on the settings in the community portal, a limitation of the provided keys per organisations can be set, same as expiration date or a revoke of the key. At this stage, all distributed keys have been limited until the end of the project duration. The CCH Key-Manager keys are not visible to an organisation and just used internally.

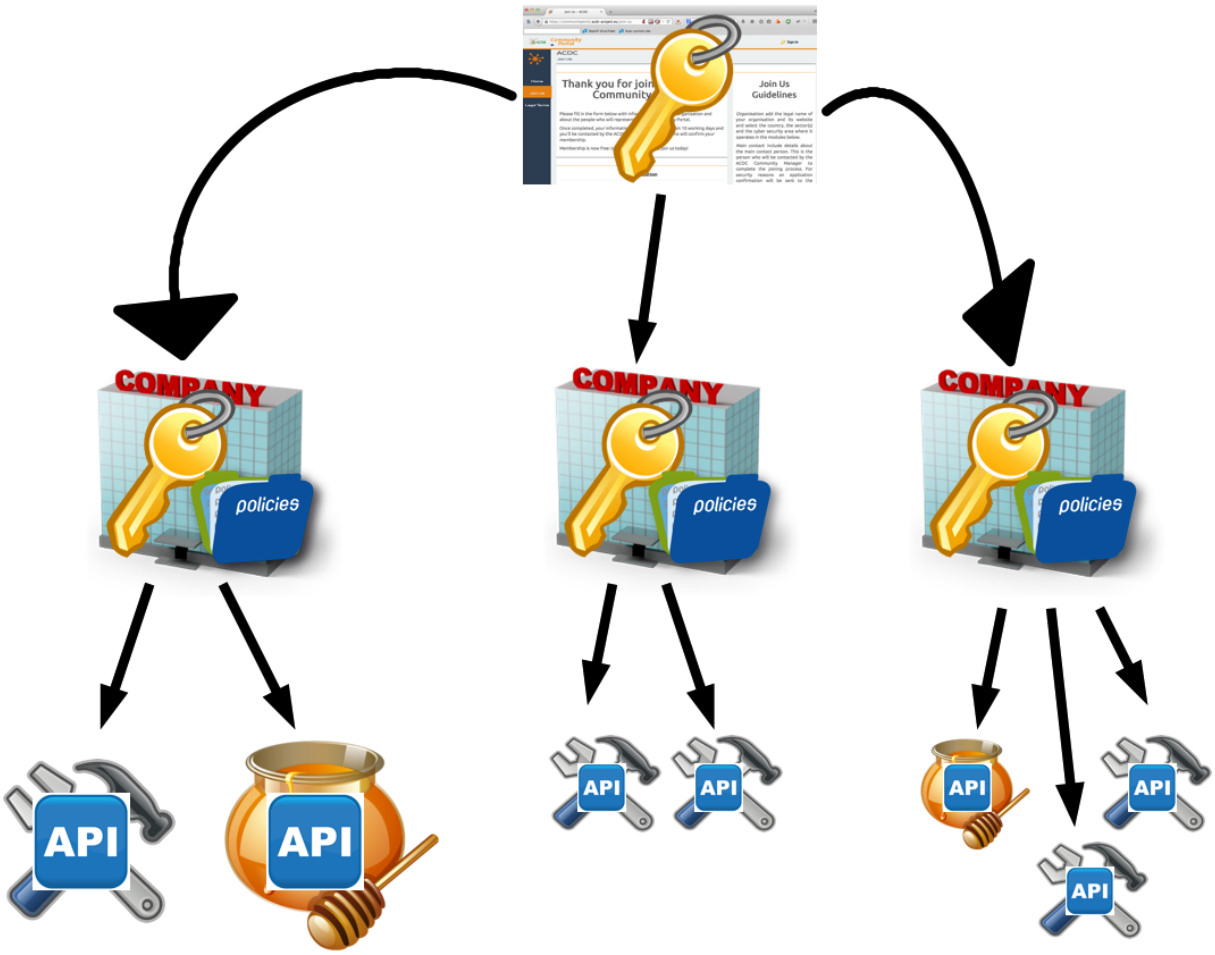


Figure 6: How API Keys are generated

Further information on the Key Management is part of deliverable **D6.2.1 ACDC Social Platform** as part of chapter 8.4

4.3.1. CCH Manager

The CCH Manager Role is the type of Admin within the key management groups.

The permission includes:

- Manage group parameters
- Add organizations to the groups / distribute CCH Key Manager Keys
- Any of the actions of other key owner groups.

4.3.2. CCH Key Manager

This organizational role allows the user to manage keys belonging to its organization. CCH Key managers can perform following actions:

- Get list of CCH API keys of the organization
- Set/unset CCH Key User role to other users of the organization
- Invalidate API keys of the CCH Key-User in the organization CCH

4.3.3. CCH Key User

A CCH Key User can perform:

- Create a new API-key for his organisation.
- Specifying, if a key is in read (data retriever) or write (data provider) modus.
- Get list of created keys grouped by read or write mode.
- Update key
- Invalidate key
- Get Key (from a recovery process)
- Manage sharing policies:
- View pending request from and to Organizations
- Approve/ Deny request of data sharing
- View list of sharing policies in place and revoke

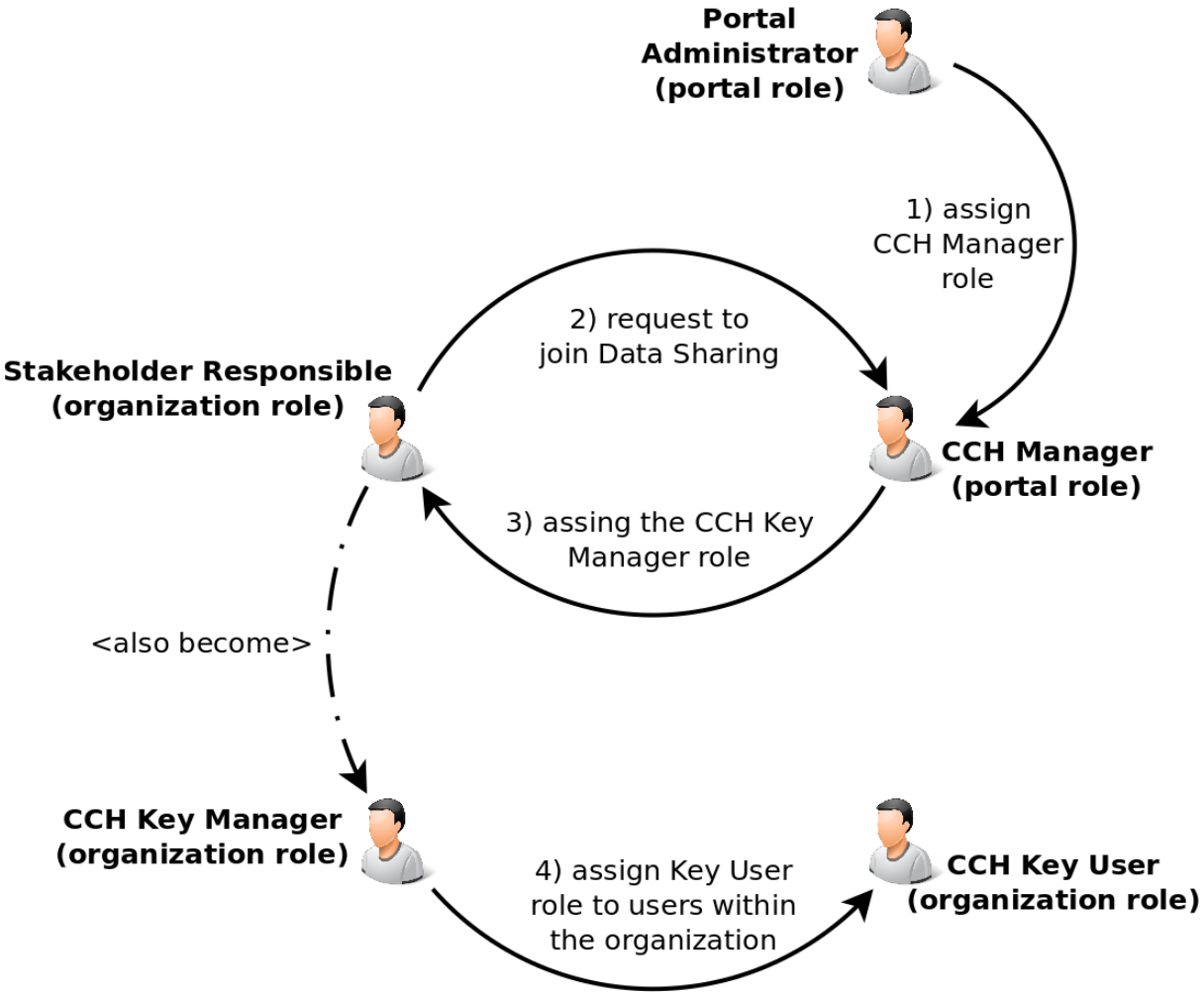


Figure 7: Key Assigment process

4.4. Examples for API-Key Management

4.4.1. Create New API Key

CreateNewApiKey - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/api_keys/
Headers	Authorization: Token token="Community_Portal_Key"
Method	POST
Content Type	application/json
Query String	-
URL	https://HOSTNAME:3000/api/v2/api_keys/
Path Params	-

data : {.....}	{
	"email":" email@community-portal.it ",
	"group_id":"[Groups]",
	"key_type":"[Key Types]",
	"description":"Testuser_Alpha"
	"data_schema_url": "http://www.schema.com/schema1"
	"super_user": [true/false],
	}
	
	Groups (group_id or group name, case sensitive!):
	/1 (or /Unverified)
	/2 (or /ISP)
	/3 (or /CERT)
	/4 (or /Antivirus)
	
	Key Types:
	- if "read" set key queries to max_queries of the group
	- if "write" set key queries to 0

Response	{ "id": 123, "key_type": "write",

```
            "group_id": "1",
            "ttl": 346,
            "description":  "My Key Description",
            "email": "email@community-portal.it",
            "created_at": "2014-11-12T12:09:13.377Z",
            "updated_at": "2014-11-12T12:09:13.377Z",
            "data_schema_url": "http://www.schema.com/schema1"
            "access_token": "YOUR_API_KEY",
            "super_user", false
}


see: curl -XPOST -H 'Authorization: Token token="Community_Portal_Key"' -H
'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/api_keys -d'{
"email":"email@community-portal.it", "group_id":1, "key_type":"write",
"description":"Testuser_Alpha", "data_schema_url":
"http://www.schema.com/schema1", "super_user":false }'
```


Table 1: CreateNewApiKey – Service

4.4.2. Update API-Key

UpdateApiKey - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/api_keys/
Headers	Authorization: Token token="Community_Portal_Key"
Method	PUT
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/api_keys/
Path Params	/ID_KEY

data : {.....}	{
	"email":" email@community-portal.it ",
	"description":"My Key Description 1",
	"ttl": [if == 0 invalidate, if >0 valid]
	}

Response	{
	"description":"My Key Description 1",
	"email": " email@community-portal.it ",
	"ttl": 0,
	"updated_at":"2014-12-15T12:09:13.377Z"
	}
	
	see: curl -XPUT -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/api_keys/147 -d'{ "email":"email2@community-portal.it", "description":"Testuser_Alpha 1"}'

Table 2: UpdateApiKey – Service

NOTE: ttl indicates the desired number of seconds the key is still valid from the time of this request. It is set to 0 to invalidate the key.

In any case, the new expiration time of the key (calculated as current time plus new ttl) cannot exceed the max expiration time of the key (calculated as key creation time plus max_ttl of the group the key belongs to). If exceeding, the new ttl will be lowered to that value.

4.4.3. Get API Key

GetApiKey Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/api_keys/
Headers	Authorization: Token token="Community_Portal_Key"
Method	GET
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/api_keys/
Path Params	/ID_KEY

Response	JSON (sample):
	{
	"access_token":"YOUR_API_KEY",
	"id":123,
	"key_type" : "write",
	"group_id" : 1,
	"ttl": 0,
	description" : "My Key Description 1",
	"email": " email@community-portal.it ",
	"created_at":"2014-11-12T12:09:13.377Z",
	"updated_at":"2014-11-12T12:09:13.377Z",
	"data_schema": "http://www.schema.com/schema1"
	}
	
	see: curl -XGET -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/api_keys/147

Table 3: GetApiKey Service

4.4.4. Replace Key

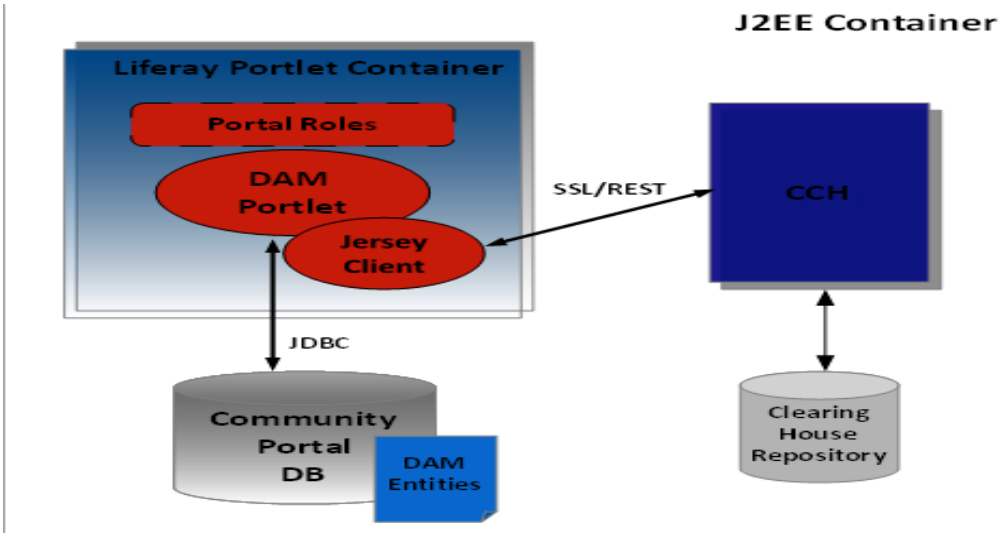
ReplaceKey Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/api_keys/replace/
Headers	Authorization: Token token="Community_Portal_Key"
Method	POST
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/api_keys/replace/
Path Params	/ID_KEY
data: {.....}	-

Response	JSON (sample):
	{
	"access_token":"YOUR_API_KEY",
	"id":124,
	"key_type" : "write",
	"group_id" : 1,
	"ttl": 0,
	"description" : "My Key Description 1",
	"email": " email@community-portal.it ",
	"created_at":"2014-11-12T12:09:13.377Z",
	"updated_at":"2014-13-12T12:09:13.377Z",
	"data_schema": "http://www.schema.com/schema1"
	}
	
	see: curl -XPOST -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/api_keys/replace/147

Table 4: ReplaceKey Service

4.5. Data Access Management (DAM)

The authentication mechanism of the CCH combines the use of a solid state-of-the-art Identity Management System (IdMS), including a verification process to approve an applicant. For this reason Trusted Introducer (TI) and the Community Portal to centralize the authentication is very useful in this context. Therefore any ACDC components connecting to the CCH must support to work in this type of IdMS environment.



Restricted access will be handled via the community portal, where API keys are generated for every data feed request. The workflows for specific stakeholders are being defined within the work related to **D1.7.2 – Data Format Specification**. The different workflows and related restrictions are covered within chapter 10 of that document. The specifications will be implemented into the CCH. The draft of **D1.7.2** currently covers CERTs, ISPs and Research. Other groups and settings will be added within further progress of the project and be based on real-time usage through internal and external ACDC stakeholders.

The *Data Access Management (DAM)* is owned and managed by the Community Portal. Further information on DAM can be retrieved from **D6.2.1 ACDC Social Platform** as part of chapter 8.4

4.6. Group Management

This service "group Management" retrieves the list of groups for the API-Keys. Each API-Key belongs to only one group, which imposes constraints on the usage of the key itself. Each group has an ID, identifying the group and its label. Currently, four groups have been foreseen for interaction with the CCH: Unverified, ISP, CERT, Antivirus.

Each organization using CCH services is associated to a group in the DAM, and API-Keys are associated to groups in the CCH.

Current constraints on the key usage associated to each group are:

- max_ttl – the maximum time to leave the key can have, when expired the key must be replaced this constraint is taken into account by the DAM when sending key creation requests to the CCH, and cross checked by the CCH on received requests.
- max_ttl is a unix_timestamp if 0 ttl is infinity
- max_ttl is in seconds that will be checked against the TTL value in the api_key request (create/update)
- max_queries – the maximum number of "read" requests that can be performed using the key. Some groups like Antivirus have the possibility to re-increment the number by performing "write queries" with other keys of their organizations. This constraint is enforced by the CCH on each "read" request.

- max_keys – The maximum number of active (i.e. not expired) keys the organization can create.
- data_types – The type of data that can be retrieved by using the key.

4.7. Examples Group Management

4.7.1. Get Groups

GetGroups - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/groups/
Headers	Authorization: Token token="Community_Portal_Key"
Method	GET
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/groups/
Path Params	-
Response	{"groups": [1, 2, 3, 4]} see: curl -XGET -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' https://HOSTNAME:3000/api/v2/groups

Table 5: GetGroups Service

4.7.2. GetGroups By ID – Service

GetGroups By ID - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/groups/
Headers	Authorization: Token token="Community_Portal_Key"
Method	GET
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/groups/
Path Params	Group id (or group name, case sensitive!): /1 (or /Unverified) /2 (or /ISP) /3 (or /CERT) /4 (or /Antivirus)

Response	{
	"group_id": 1,
	"label": "Unverified",
	"max_ttl":0,
	"max_queries": 10,
	"max_keys": 10,
	"data_types":
	[
	{
	"data_type_id": 1,
	"label": "IP Address"
	}
	]
	}
	
	see:
	curl -XGET -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/groups/1

Table 6: GetGroups by ID

4.7.3. Update Group – Service

This service updates the parameters associated to a given group. Only the set of parameters included in the PUT request will be updated for the group, other parameters will be kept as they are. Usage of this service must be restricted to super-users of the CCH only. The table below contains parameters for the implementation of the service.

UpdateGroup - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/groups/
Headers	Authorization: Token token="Community_Portal_Key"
Method	PUT
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/groups/ISP
Path Params	Group id (or group name, case sensitive!): /1 (or /Unverified) /2 (or /ISP) /3 (or /CERT) /4 (or /Antivirus) ...

data : {.....}	JSON:
	{
	"max_ttl": 0,
	"max_queries": 50,
	"max_keys": 10,
	}

Response	JSON:
	{
	"label": "ISP",
	"max_ttl": 0,
	"max_queries": 50,
	"max_keys": 10,
	"data_types": "[{"data_type_id": 1, "label": "IP Address"}, …..]"
	}
	see: curl -XPUT -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' –k https://HOSTNAME:3000/api/v2/groups/Unverified -d '{"max_queries": 50}'

Table 7: Update Group Service

4.7.4. CreateGroup - Service

This service creates a new group. All parameters for the group are mandatory. The CCH returns error in case a group with the same label already exists. Upon creation, CCH returns the ID of the new group created, and parameters of the new group. Usage of this service must be restricted to CCH-Key-User of the CCH only. The table below contains parameters for the implementation of the service.

CreateGroup - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/groups/
Headers	Authorization: Token token="Community_Portal_Key"
Method	POST
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/groups/ISP
Path Params	-

data : {.....}	{
	"label": "New Group",
	"max_ttl": 0,
	"max_queries": 10,
	"max_keys": 10,
	"data_types": [1,2,3] <- no brackets arround an array!!
	}

Response	{
	"group_id": 5,
	"label": "New Group",
	"max_ttl": 0,
	"max_queries": 10,
	"max_keys": 10,
	"data_types": "[{"data_type_id": 1, "label": "IP Address"}, …..]"
	}
	see: curl -XPOST -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/groups -d '{ "label": "New Group", "max_ttl": 0, "max_queries": 10, "max_keys": 10, "data_types": [1,2,3] }'

Table 8: CreateGroup - Service

4.7.5. Delete Group

This service deletes an existing group. When a group is deleted, all the API-Keys associated to that group in the CCH will be associated to the unverified group, which cannot be deleted or modified. Usage of this service will be restricted to CCH Key-user of the CCH only. The table below contains parameters for the implementation of the service.

DeleteGroup - Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/groups/
Headers	Authorization: Token token="Community_Portal_Key"
Method	DELETE
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/groups/ISP
Path Params	Group id (or group name, case sensitive!): /2 (or /ISP) /3 (or /CERT) /4 (or /Antivirus) ...

Response	JSON:
	{
	"result":"[true|false]"
	}
	
	see:
	curl -XDELETE -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/groups/6

Table 9: DeleteGroup – Service

5. Data Sharing Policies

The "Sharing Policies" module of the Data Access Management (DAM) in the community portal will use sharing Policies management services exposed by the CCH. These services enable the management of data sharing policies between organizations through the CCH services.

5.1. Add Sharing Policy

This service adds a new sharing policy between two API-Keys. One key has read-only access, while the other has both (read/write). Both keys can be used to setup the sharing policy. The CCH service will detect the channel direction based on the key_type properties of the two keys identifiers.

The table below contains parameters for the implementation of the service.

AddSharingPolicy Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/sharing_policies/
Headers	Authorization: Token token="Community_Portal_Key"
Method	POST
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/sharing_policies/
Path Params	/{int: ID_KEY}

data: {.....}	JSON (sample):
	{
	"other_key_id": 234
	}

Response	JSON (sample):
	{
	"result":"[true|false]"
	}
	
	see: curl -XPOST -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/sharing_policies/147 -d '{"other_key_id":145}'

Table 10: AddSharingPolicy Service

5.2. Delete Sharing Policy

This service deletes an existing sharing policy. The request can be sent by two keys, one key has read-only access, while the other has both (read/write). In both cases the effect is the immediate ending of the data flow between the read and the write keys.

The table below contains parameters for the implementation of the service.

DeleteSharingPolicy Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/sharing_policies/
Headers	Authorization: Token token="Community_Portal_Key"
Method	DELETE
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/sharing_policies/
Path Params	/{int: ID_KEY}

data: {.....}	JSON (sample):
	{
	"other_key_id": 234
	}

Response	JSON (sample):
	{
	"result":"[true|false]"
	}
	
	see: curl -XDELETE -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/sharing_policies/147 -d '{"other_key_id":145}'

Table 11: DeleteSharingPolicy Service

5.3. Get Sharing Policies

This service allows the user to retrieve the list of sharing policies attached to the keys.

The table below contains parameters for the implementation of the service.

GetSharingPolicy Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/sharing_policies/
Headers	Authorization: Token token="Community_Portal_Key"
Method	GET
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/sharing_policies/
Path Params	-

data: {.....}	JSON (sample):
	{
	"email": " email@community-portal.it "
	}

Response	JSON (sample):
	{
	"channels":
	[
	{"myKey": 255, "stackholders_key" : 322},
	{"myKey": 266, "stackholders_key" : 456},
	{"myKey": 278, "stackholders_key" : 521},
	]
	}
	
	see: curl -XGET -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/sharing_policies/ -d '{"email":" email@community-portal.it "}'

Table 12: GetSharing Policy - Service

6. Data & Data Formats

The definition and specification on data formats and their retrieval and standards is part of deliverable **D1.7.2 Data Formats.** The lecture of this document is recommended as an addition to this chapter.

6.1. Schemata

As part of the pre-work and assessment by the project partners, it has been determined that a basic standardisation of the submitted data fields and basic requirement on mandatory fields simplifies the data submission and retrieval. These specifications have been defined as the ACDC - "Schemata". These have been outlined and defined in the Deliverables **D1.7.1/2 "Data Formats Specification**".

Schemata are being worked out in **D1.7.2 Data Formats** and will be managed by Community Platform. The Community Platform will provide storage for the different schemata and will distribute them to every interested ACDC Partner.

The current process to submit a schemata proposal is handled manually. A partner involved in WP2 makes a suggestion for schemata, which gets verified and approved by the partners of WP1.

The ACDC roadmap includes an automated schemata upload through the community portal, which includes a verification and automated adaption into the CCH.

6.2. Example Schemata: Submission of a Malware Sample

Example how to submit a Malware sample (from D.1.7.2, chapter 7.3.7)

This report is used to submit a malware sample to the CCH. The associated report_category is *eu.acdc.malware.*

Subcategory
There are no subcategories for the eu.acdc.malware report.

Schema

Malware Sample – eu.acdc.malware		
A sample of a malware.		
Required fields		
report_category	string enum: eu.acdc.malware	The category of the report: a malware sample.
report_type	string	The type of the report. This is a free text field characterising the report that should be used for a human readable description rather than for automatic processing. As a rule of thumb this should not be longer than one sentence.
timestamp	string format: date-time	The timestamp when the sample was obtained.
source_key	string enum: malware	The type of the reported object: a malware sample.
source_value	string	The SHA256 hash of the malware sample.
confidence_level	number minimum: 0.0 maximum: 1.0	The level of confidence put into the accuracy of the report. A number between 0.0 and 1.0 with 0.0 being unreliable and 1.0 being verified to be accurate.
version	integer enum: 1	The version number of the data format used for the report.
Optional fields		

report_id	integer	The ID of the report in the CCH. This will be set by the CCH and is thus overwritten on import.
reported_at	string format: date-time	The timestamp when the report was submitted to the CCH. This will be set by the CCH and is thus overwritten on import.
botnet	string	The botnet the sample is attributed to. This reference a separate eu.acdc.botnet report.
additional_data	object	Additional data for the observation. This allows putting more specific information into a report on a case by case basis in a structured manner. The usage of this field is at the data providers discretion.
alternate_format_type	string enum: IDMEF, STIX	The type of the alternate format description of the observation.
alternate_format	string requires: alternate_format_type	A description of the observation in an alternate format. This is used to submit complex structured formats like IDMEF to the CCH.
sample_b64	string	The source code of the sample encoded in Base64.
mime_type	string	The MIME type of the sample.
sample_hashes	array items: object(hash function, hash value)	An array of objects containing hashes for the sample. Each item gives a hash function and the corresponding hash value.
exploits	array items: object(identifier scheme, exploit identifier)	Exploits discovered in the analysed sample. This is an array of objects, each giving an identifier scheme like CVE and an identifier for the actual exploit found.
Dependencies		
If the report contains an alternate_format, it has to specify the alternate_format_type as well.		

A malware captured by a honeypot might be submitted to the CCH in the following manner. The sample_b64 is truncated here but SHALL be complete when submitting a sample.


```
{
        "report_category": "eu.acdc.malware",
        "report_type": "Bot from honeypot capture",
        "timestamp": "2014-06-15T15:47:12Z",
        "source_key": "malware",
        "source_value":
"933ff380eb1448c5c583796fdc6ce842eec14a23b686fff4672c85a7ee9d7d0a",
        "sample_b64":
"ewogICAgInRpdGxlIjogIk1hbHdhcmUgU2FtcGxlIiwgCiAgICAiZGVzY3JpcHRpb24iOiAK...",
        "confidence_level": 1.0,
        "version": 1
}
```

6.3. Minimal Dataset

The CCH accepts a basic set of commands with which external sources can report security incidents.

ACDC Minimal dataset

This is the minimal data set requirement to submit and to receive data from the CCH.

Required fields		
report_category	string	The category of the report. This links the report to one of ACDC's schemata. A report category has the format 'eu.acdc.<identifier>'.
report_type	string	The type of the report. This is a free text field characterising the report that should be used for a human readable description rather than for automatic processing. As a rule of thumb this should not be longer than one sentence.
timestamp	string format: date-time	The timestamp when the reported observation took place. This can for example be when an attack occurred, when a malware hosting was observed, or when a compromise took place according to log files.
source_key	string enum: ip, malware, subject, uri	The type of the reported object.
source_value	string	The identifier of the reported object like its IP address or URI.
confidence_level	number minimum: 0.0 maximum: 1.0	The level of confidence put into the accuracy of the report. A number between 0.0 and 1.0 with 0.0 being unreliable and 1.0 being verified to be accurate.
version	integer enum: 1	The version number of the data format used for the report.
Optional fields		
report_id	integer	The ID of the report in the CCH. This will be set by the CCH and is thus overwritten on import.
reported_at	string format: date-time	The timestamp when the report was submitted to the CCH. This will be set by the CCH and is thus overwritten on import.
botnet	string	The botnet the observation is attributed to. This can for example be the botnet a malware joins, the botnet that sends a spam campaign, or the botnet a bot belongs to.

Figure 8:ACDC Minimal Dataset

Additional requirements and advanced schemes have been defined by WP1. It is documented as the "Schemes Proposal" in **Deliverable 1.7.2 Data Format** in chapter 6 and following.

6.4. Real-Time Access / Channels

The following chart illustrates the concept of the data flow to an authorised stakeholder from the community portal, the assignments of policies based on his permissions and interest, into the CCH and the channels he is authorised to receive his data.

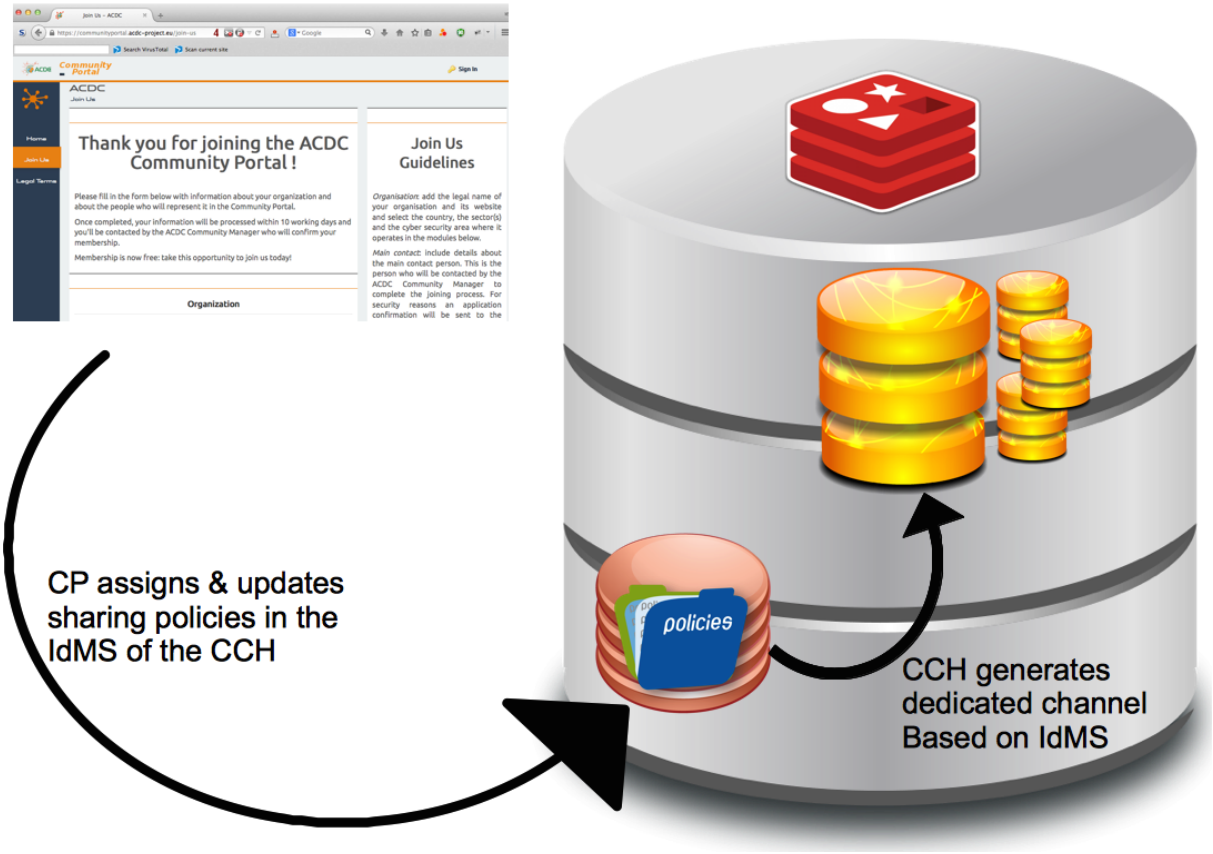


Figure 9: Channels for Real-Time Access

6.5. Get All Data Channels

This service retrieves the list of all channels (read or write) for data sharing; each channel corresponds to an API-Key.

The operation returns all the channels (other API-keys) that can be associated to the API-Key indicated in the request path. The returned key depends on the type of the key indicated:

- If the API-key is a read key, then all the Write-Keys IDs are returned
- If the API-key is a write key, then all the Read-Keys IDs are returned

The authorization token of a CCH Master-Key must be provided to perform this operation.

The table below contains parameters for the implementation of the service.

GetAllDataChannels Service	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v2/
Service Name	/channels/
Headers	Authorization: Token token="Community_Portal_Key"
Method	GET
Content Type	application/json
QueryString	-
URL	https://HOSTNAME:3000/api/v2/channels/
Path Params	/ID_KEY (read or write key)
data: {.....}	-

Response	JSON (sample):
	{
	"keys":
	[
	{"id": 255, "email": "email 1", "description": "Test User 1"},
	{"id": 324, "email": "email 2", "description": "Test User 2"},
	{"id": 13, "email": "email 3", "description": "Test User 3"}
	]
	}
	
	see: curl -XGET -H 'Authorization: Token token="Community_Portal_Key"' -H 'Content-Type: application/json' -k https://HOSTNAME:3000/api/v2/channels/145

Table 13: GetAllDataChannels Service

6.6. Output Formats and example

The CCH offers different output formats to present the results of a query:

- JSON (Default)
- XML
- YAML

Example: query for data about ASN 8560, output format text/YAML.

query for data about ASN 8560	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/asns/AS8560
Headers	Authorization: Token token="Your_API_key"
Method	GET
Accept	text/yaml
QueryString	-
URL	https://HOSTNAME:3000/api/v1/asns/
Path Params	AS8560
Response	YAML See: curl -k -XGET https://HOSTNAME:3000/api/v1/asns/AS8560 -H 'Authorization: Token token="Your_API_key"' -H "Accept: text/yaml"

Table 14: query for data about ASN 8560

Example: query for data about ASN 8560, output format XML

query for data about ASN 8560	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/asns/
Headers	Authorization: Token token="Your_API_key"
Method	GET
Accept	text/xml
QueryString	-
URL	https://HOSTNAME:3000/api/v1/asns/
Path Params	AS8560
Response	XML See: curl -k -XGET https://HOSTNAME:3000/api/v1/asns/AS8560 -H 'Authorization: Token token="Your_API_key"' -H "Accept: text/xml"

Table 15: query for data about ASN 8560

6.7. How to submit a (domain) report:

This example describes how to report the domain "heise.de" as malicious. The report type here has been defined as "Malware_Test". The TTL is set to 1800 seconds.

Submit a report	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/reports/
Headers	Authorization: Token token="Your_API_key"
Method	GET
Content Type	application/json
QueryString	-

Data	{
	"report_type":"Malware_Test",
	"source_key":"domain",
	"source_value":"heise.de",
	"reported_at":1401717657,
	"ttl":1800
	}

URL	https://HOSTNAME:3000/api/v1/reports/
Path Params	-
Response	{ "id":21991337, "api_key":"your api key", "reported_at":"2014-06-02T14:00:57.000Z", "report_type":"Micha Test", "description":null, "ttl":1800, "source_id":149665, "source_type":"Domain", "solved":false, "abuse_reported":null, "job_id":"523d5bd7c1dd7bb28b044a52", "created_at":"2014-06-02T14:14:47.884Z",

	"updated_at":"2014-06-02T14:14:47.920Z" } See: curl -k -q -XPOST https://HOSTNAME:3000/api/v1/reports/ -d '{"report_type":"Micha Test","source_key":"domain","source_value":"heise.de","reported_at":1401717657,"ttl":1800}' -H 'Authorization: Token token="your api key"' -H 'Content-Type:application/json'

Table 16: submit a report

6.8. Query for a domain name

Query for domain name	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/domains/
Headers	Authorization: Token token="Your_API_key"
Method	GET
Content Type	application/json
QueryString	-

Data	{
	"all":1
	}

URL	https://HOSTNAME:3000/api/v1/domains/
Path Params	heise.de

Response	{
	"incidents":{"incident_types":{}},
	"source":"heise.de",
	"last_seen":null,
	"first_seen":null,
	"report_count":1,
	"count_seen":0
	}
	
	See: curl -k -q -XGET https://HOSTNAME:3000/api/v1/domains/heise.de -H 'Authorization: Token token="your api key"' -H 'Content-Type: application/json' -d '{"all":1}'

Table 17: Ask for domain name

Query for domain name	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/domains/
Headers	Authorization: Token token="Your_API_key"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/domains/
Path Params	heise.de
Response	{ "incidents": { "incident_types": { "Malware_Test": [{ "source":"heise.de", "report_id":21991337 }] } }, "source":"heise.de", "last_seen":"2014-06-02T14:14:47.000Z", "first_seen":"2014-06-02T14:14:47.000Z", "report_count":1, "count_seen":1 } See: curl -k -q -XGET https://HOSTNAME:3000/api/v1/domains/heise.de -H 'Authorization: Token token="your api key"' -H 'Content-Type: application/json'

Table 18: Ask for domain name

6.9. Query Incident ID

Query for the ID of this incident	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/reports/
Headers	Authorization: Token token="Your_API_key"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/reports/
Path Params	21991337

Response	{
	"id":21991337,
	"api_key":"your api key",
	"reported_at":"2014-06-02T14:00:57.000Z",
	"report_type":"Malware_Test",
	"description":null,
	"ttl":1800,
	"source_id":149665,
	"source_type":"Domain",
	"solved":false,
	"abuse_reported":null,
	"job_id":"523d5bd7c1dd7bb28b044a52",
	"created_at":"2014-06-02T14:14:47.000Z",
	"updated_at":"2014-06-02T14:14:47.000Z"
	}
	
	See: curl -k -q -XGET https://HOSTNAME:3000/ api/v1/reports/21991337 -H 'Authorization: Token token="your api key"' -H 'Content-Type: application/json'

Table 19: Ask for the ID of this incident

6.10. Retrieve a file from the CCH

Retrieve a file from a CCH	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/uploads/
Headers	Authorization: Token token="Your_API_key"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/uploads/
Path Params	/malware/6261/e59ea3a6-006a-437f-94d5-5abecb361cd8
Response	Downloads the file with the name: aae0f083745d16fe487c26844a50fa1c See: curl -k -J -O -v -H 'Authorization: Token token="your api key "' https://HOSTNAME:3000/api/v1/uploads/malware/6261/e59ea3a6-006a-437f-94d5-5abecb361cd8

Table 20: Downloading a file

7. Basic rights management for API keys:

The concept of the basic rights information implementation of the Community Platform covers issues like the control of own data and securing it from access of an unrestricted third party.

This following command creates an API-Key for the username „eco_test" and assigns this API key to the ASN AS8560, the AS7411 and the IP-Range 10.0.0.1/8

Assign an IP range or ASN for a given API key	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/api_keys/
Headers	Authorization: Token token="SUPER_USER_API_KEY"
Method	POST
Content Type	application/json
QueryString	-

Data	{
	"user":" eco_test",
	"email":"eMail@eco.de",
	"asns":["AS8560", "AS7411"],
	"ips":["10.0.0.1/8"]
	}

URL	https://HOSTNAME:3000/api/v1/ api_keys /
Path Params	/malware/6261/e59ea3a6-006a-437f-94d5-5abecb361cd8
Response	{ "id":140, "access_token":"3075e62dd947b4f8ddcaca4109aadec6", "ttl":1404906333, "user":"eco_test", "email":"eMail@eco.de", "superuser":false, "created_at":"2014-07-09 10:45:33 UTC", "updated_at":"2014-07-09 10:45:33 UTC", "asns": [

```
                {"id":49,"api_key_id":140,"asn":"AS7411"},
                {"id":48,"api_key_id":140,"asn":"AS8560"}
        ],
        "ips":
        [
                {
                        "id":17,
                        "api_key_id":140,
                        "ip_range":"10.0.0.1/8",
                        "first_ip":167772160,
                        "last_ip":184549375
                }
        ]
}
```


See: curl -k -v -XPOST -H 'Authorization: Token token="SUPER_USER_API_KEY"' -H "Content-Type: application/json" https://HOSTNAME:3000/api/v1/api_keys -d '{"user":"eco_test","email":"eMail@eco.de", "asns":["AS8560", "AS7411"], "ips":["10.0.0.1/8"]}'

Table 21: Assign an API key to a ASN

7.1. Testcase: Incidents from an IP (allowed IP)

Query all incidents from IP 10.0.0.2 - allowed IP -	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/ips/
Headers	Authorization: Token token="YOUR_API_KEY"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/ips/
Path Params	/10.0.0.2

Response	{
	"incidents":{"incident_types":{}},
	"source":"10.0.0.2",
	"last_seen":null,
	"first_seen":null,
	"report_count":0,
	"count_seen":0
	}
	
	See: curl -k -v -XGET -H 'Authorization: Token token="YOUR_API_KEY"' -H "Content-Type: application/json" https://HOSTNAME:3000/api/v1/ips/10.0.0.2

Table 22: show all incidents from IP 10.0.0.2 - allowed IP -

7.2. Testcase: Incidents from an IP (forbidden IP)

Query all data from IP 8.8.8.8 - forbidden IP -	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/ips/
Headers	Authorization: Token token="ANOTHER_API_KEY"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/ips/
Path Params	/10.0.0.2
Response	HTTP Token: Access denied. -> This API key is not allowed to ask for other IPs See: curl -k -v -XGET -H 'Authorization: Token token="ANOTHER_API_KEY"' -H "Content-Type: application/json" https://HOSTNAME:3000/api/v1/ips/8.8.8.8

Table 23: show all data from IP 8.8.8.8 - forbidden IP -

7.3. Testcase: Incidents from an ASN (allowed IP)

Query for the ASN 7411 (with permission)	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/asns/
Headers	Authorization: Token token="YOUR_API_KEY"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/asns/
Path Params	/AS7411

Response	{
	"incidents":{"incident_types":{}},
	"source":"AS7411",
	"last_seen":null,
	"first_seen":null,
	"count_seen":0
	}
	
	See: curl -k -v -XGET -H 'Authorization: Token token="YOUR_API_KEY"' -H "Content-Type: application/json" https://HOSTNAME:3000/api/v1/asns/AS7411

Table 24: Asking for the ASN 7411 is allowed as stated above

7.4. Testcase: Incidents from an ASN (forbidden IP)

Query for the ASN 4711 ID denied by the CCH	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/asns/
Headers	Authorization: Token token="ANOTHER_API_KEY"
Method	GET
Content Type	application/json
QueryString	-
Data	-
URL	https://HOSTNAME:3000/api/v1/asns/
Path Params	/AS4711
Response	HTTP Token: Access denied. -> This API Key is not allowed to see other ASNs See: curl -k -v -XGET –H 'Authorization: Token token="ANOTHER_API_KEY"' -H "Content-Type: application/json" https://HOSTNAME:3000/api/v1/asns/AS4711

Table 25: Asking for the ASN 4711 id denied by the CCH

7.5. Testcase: Array Overwriting

Overwrite the Arrays for ASN and IP by PUT in new ones	
Protocol	SSL
Host	HOSTNAME
Port	3000
Base URL	/api/v1/
Service Name	/api_keys/
Headers	Authorization: Token token="SUPER_USER_API_KEY"
Method	PUT
Content Type	application/json
QueryString	-
Data	{

	"asns":["AS4711"], "ips":["10.0.0.1/21"] }
URL	https://HOSTNAME:3000/api/v1/api_keys/
Path Params	/API_KEY

Response	{
	"id":140,
	"access_token":"3075e62dd947b4f8ddcaca4109aadec6",
	"ttl":1404906333,
	"user":"eco_test",
	"email":"eMail@eco.de",
	"superuser":false,
	"created_at":"2014-07-09 10:45:33 UTC",
	"updated_at":"2014-07-09 10:45:33 UTC",
	"asns":
	[
	{
	"id":50,
	"api_key_id":140,
	"asn":"AS4711"
	}
	],
	"ips":
	[
	{
	"id":18,
	"api_key_id":140,
	"ip_range":"10.0.0.1/21",
	"first_ip":167772160,
	"last_ip":167774207
	}
	]
	}
	
	See: curl -k -v -XPUT -H 'Authorization: Token token=" SUPER_USER_API_KEY "' -H "Content-Type: application/json" https://HOSTNAME:3000/api/v1/api_keys/API_KEY
	-d '{"asns":["AS4711"], "ips":["10.0.0.1/21"]}'

Table 26: Overwrite the Arryas for ASN and IP by PUT in new ones