White Paper, First Draft: Access Control for Smart Home: a Specification of Interactive Policy Administration Point*

No Author Given

No Institute Given

Abstract. Policy Administration Point (PAP) module, a part of access control system, allows user to manage and create access control policies to be used to control access to smart devices and data pertaining to them. In this paper, we define the PAP specification to be used in different smart home system architecture and configuration. This specification is designed to support different access control models and policy expression languages. Two different methods for policies creation request are also considered in this specification: manual text and voice request (e.g. voice search or command for policies management).

Keywords: Policy Administration Point; access control; smart home; IoT; security; smart home hub;

1 Introduction

In general, smart home devices are connected to smart home hub, through which user can control and access them. The smart home hubs, existed in the market, are built by different companies (e.g. samsung smartthing, logitech harmony, etc.) [16] based on different access control specifications: different access control models and policy expression languages.

Each designed hub supports a limited number of smart devices as defined by the company that creates the hub. Thus, it is possible that the same smart home system may need to use hubs from different companies if the provided functionalities of one hub is not enough to address the required needs in the home system. Using multiple hubs of different brands in the same smart home system may not be an easy task since each hub provides its own device management console. Using different console to manage devices in a home is not a user friendly system, hence, harmonising the device management console for different hubs can facilitate user in managing those smart devices and make smart home system more attractive. In this papers, we introduce a PAP specification supporting different access control models and policy expression languages. The aim is to centralise the device management console (PAP in particular) for different hubs used in the smart home system. The PAP system, built based on this specification, is

* This work is supported by FEDER-IDEE Project.

able to create different access control policies based on user's preference access control model and policy expression languages. Three access control models are discussed in this paper: attribute-based Access Control (ABAC), Role-based Access Control (RBAC) [7] and Adaptive Risk-aware Access Control [12]. For policy language, we focus on eXtensible Access Control Markup Language (XACML) [11] and OneM2M's specification for access control and authorisation [17]. This specification focuses also on two different methods for policies creation request: by manual text and by voice request.

This paper is structured as follows. Section II is about the global architecture of smart home system where the PAP is integrated in. Section III presents the functional requirements and detailed global architecture of PAP. Section IV talks about the PAP communication message format. Section V dedicates to the PAP client interface design and Section VI is the conclusion.

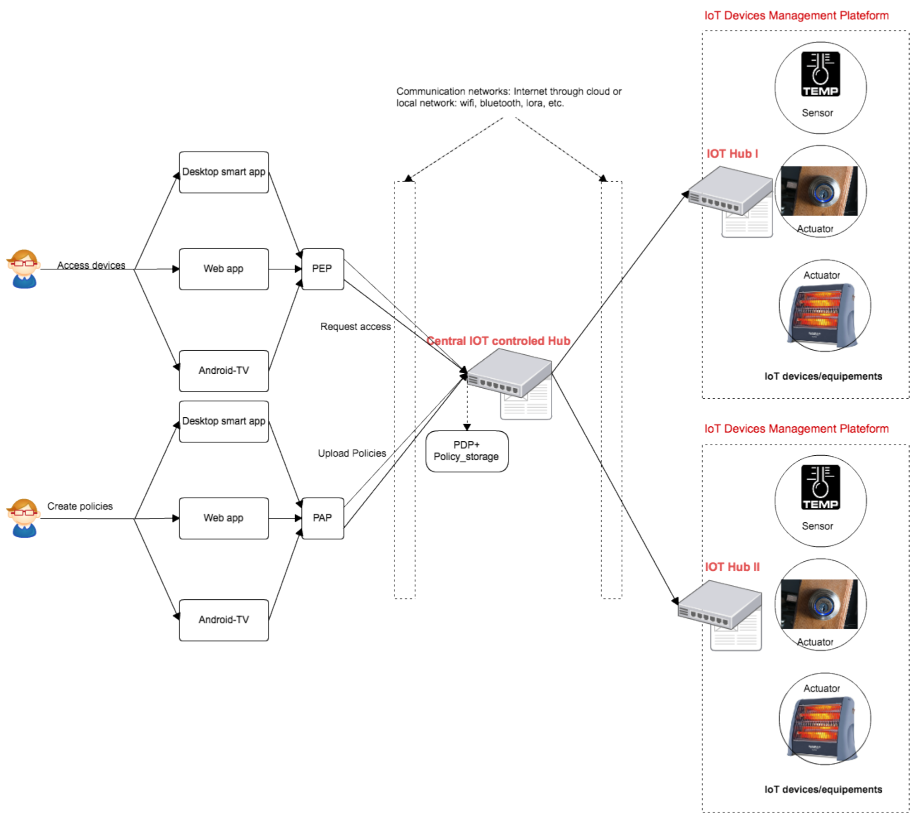


Fig. 1. Smart home architecture with access control management and decision modules

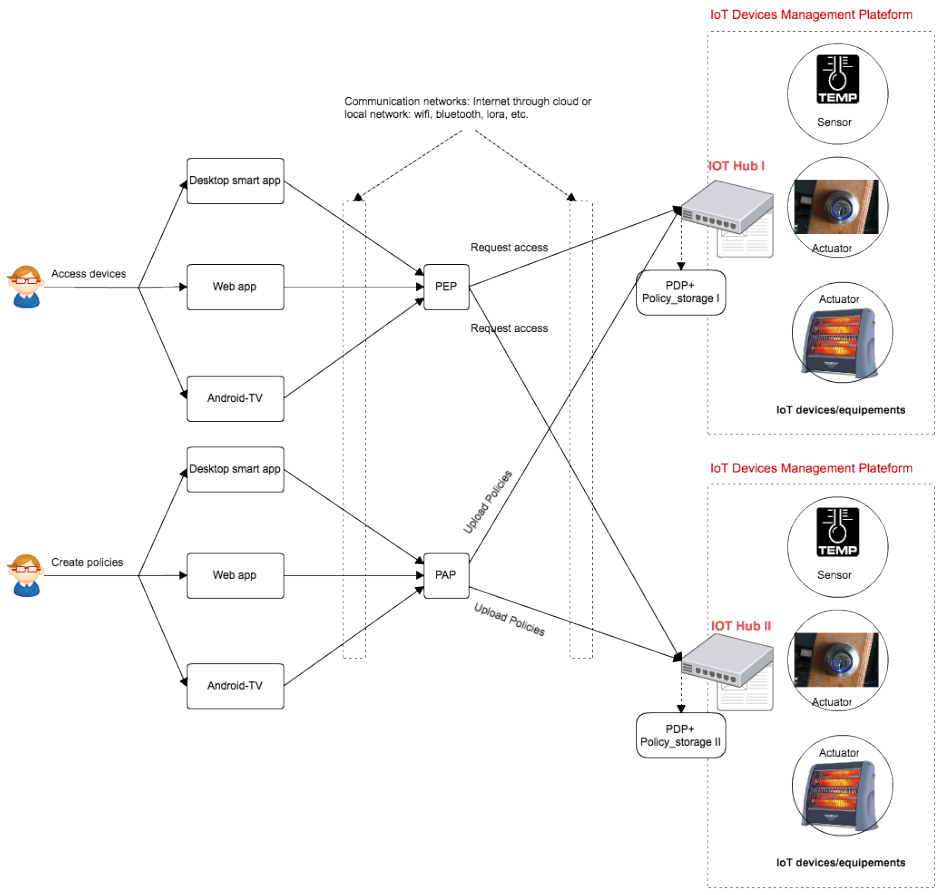


Fig. 2. Smart home architecture with access control management and decision modules

2 Global Architecture: smart home system with PAP integration

Smart home system is a new technology and it not mature yet [12], most of the existing technologies [16] provides basic smart home functionalities [8] where security of data and devices are basically based on the traditional authentication method (e.g. username and password). For example, smartthing hub [16] developed by Samsung [16] allows users to control devices by mean of user account in Samsung cloud service. The current version of smartthing does not support any customised access control policies definition to reflex user's preference. However, since smart home system is evolving, the future system may provide more options for user to control their devices. Taking into account different aspects of the existing technologies and the future evolution that they might have, we define the global architecture of smart home system where access control management and decision modules are integrated as in Figure 1 and Figure 2. In both figures, there are the same functional modules, only their positions are different. Those modules are:

– IoT device management platform is a network of smart devices controlled by the IoT hub. The devices can be anything such as environmental sensors, actuator, or other type of smart devices.
– IoT smart home Hub, a smart component, allows all devices in the smart home network to securely communicate with each other. This component is also responsible for communicating with IoT service in the cloud through different network technologies for information exchange.
– PDP (Policy Decision Point) is the module responsible for evaluating the access control policies defined by user.
– Policy Storage is a media storage storing access control policies uploaded by PAP. This policy storage is accessible by PDP during policy evaluation phrase.
– PAP is the policy administration module, a console allowing user to define access control policies for devices in the network. Once the policy is created, PAP uploads policy to policy storage.
– PEP (policy enforcement point) is responsible for enforcing the extra obligation that user or system may need to perform before or after the access permission is granted.
– Desktop smart app/ web app/ Android TV are the different types of user's media used to access different smart home modules, such as accessing and managing devices.
– Centralised IoT controlled hub (Figure 2) is a virtual smart home hub used as interface between PEP, PAP and the smart home hubs (hubs produced by different companies).

We propose two different architectures based on two assumptions. Firstly, in case the existing smart home hub does not support access control mechanism and it uses only the traditional authentication method for securing devices and data pertaining to them. Secondly, we assume that the smart home hub is able to perform the policy decision and has some memory for policies storage. These two assumptions lead to two different architectures proposed in Figure 1 and Figure 2 respectively.

1. Architecture 1 (Figure 1). In smart home context, a fine-grained access control policy is required in order to limit access to different types of user it may have. For example, CCTV can be controlled only by parents and children with certain ages are prohibited to access some TV channels that are not suited for them. Another example, controlling smart door remotely may be prohibited from child given the risk he may create. Given these examples, we can see clearly that there is a need for a fine-gained and different level of access control to smart devices, the traditional authentication is not enough to address this issue. However, most existing smart home systems, such as smartthing hub [16], logitech harmony [18], google hub [19] do not provide to user the ability to define the complex access control policies for smart devices in home. They use a simple authorisation method by means of user's account to control access to device. To provide that abilities for user to define his own complex access control policies, we need to introduce another module acting as the intermediate entity (central IoT control hub) between the smart home hub and user's application. We propose the smart home architecture with the access control system as presented in Figure 1. In that architecture, every smart home hub[1] in the system is connected to the central IoT control hub. Central hub accesses devices through different hubs by mean of authentication method. The access control decision is performed at central hub. User in the system connects to devices through central hub instead of each hub in the sub network of devices. As shown in Figure 1, user can access device or create access control policies. When accessing smart device, user's access request needs to be sent to PEP module, which forwards the request to PDP module that is generally integrated in the central hub. If the PDP provides positive response, the central hub will connect to subnetwork's hub and build a secure channel where user can use later to access devices in that network. In the architecture in Figure 1, user can also define the access control polices, through PAP module. Once the policy is created, PAP uploads the policy to central hub where it is stored.
2. Architecture 2 (Figure 2). In this architecture, we assume that each smart home hub has the capability to evaluate access control policy. In other words, the access control system is integrated and pre-installed in the hub. Different companies, building smart home hubs, may use different access control model and policy languages for expressing access rules. Thus, central PAP module, as in Figure 2, should be able to support the policy expression for different access control models and policy languages. In system architecture in Figure 2, user can create access control policies using PAP modules, when creating policies, user can choose the access control model he prefers and

[1] This smart home hub is the hub produced by different companies, such as Samsung smartthing, logitech harmony, google hub, etc.

policy language used to express the rule derived from the chosen model. The chosen access control model and policy language should be compatible with the model and language used in each smart home hub. Once the policy is created, PAP uploads the policy to the chosen smart home hub where it is stored. The created policy will later be used for controlling the access to smart devices in the network. In this architecture, when user wants to access device, user needs to send access request to PEP module where the request is further forwarded to the corresponding smart home hub where the requested device is situated. The access decision to whether allow or deny access is performed at hub.

The multiple access control models and policy languages supporting PAP module is one of the important components in both architectures. Thus, in this paper, we focus on defining the detailed specification of PAP module. This includes, the PAP architecture, communication protocol, communication message, policy languages and access control models.

3 Functional requirements and Global Architecture of PAP

In this section, we focus on the functional requirements and PAP's architecture. The structure of request message and policy language interpretation are also presented in this section.

3.1 PAP Functional requirements

The main objective of building PAP is to facilitate user in devices and access control policies management. Moreover, the future PAP system should help centralise the devices and policy management in case smart home hubs, from different brands, are used. Furthermore, user friendly system is also our aim, hence, we also introduce the requirements for voice request and voice search in this specification. Below are the PAP functional requirements for smart home system.

1. Usability. User should be able to perform necessary operations concerning the policy management task. The following operations should be available for user.
   - Create policy. User should be able to create different access control polices with constraints and conditions.
   - Delete policy. In case user wants to delete policy, he should be able to do so.
   - Modify policy. Policy can be changed, hence, it is imperative that user should be allowed to modify policy to his wishes.
2. Policy expression. Given the heterogeneity of access control policies used in smart home hubs, it is important that user is able to choose different policy languages to express his access control rule.

3. Access control model. Different smart home hub manufacturers may adapt different access control models, hence, it is imperative that multiple access control models should be available for user to choose.
4. User's preference setting. Future PAP system should allow user to set his preference system's setting, such as, different type of access control models, policy languages, communication protocol between PAP and hubs, messaging protocol between PAP and hubs and the security feature.
5. Extensible. The PAP system should be extensible, in the sense that if user wants to add new access control model and policy languages, it should be possible.
6. User friendly and smart. To make smart home smarter, user should be provided a more user friendly tools for managing his devices. Creating and searching access control policy by voice are some of the features that should be available to user.

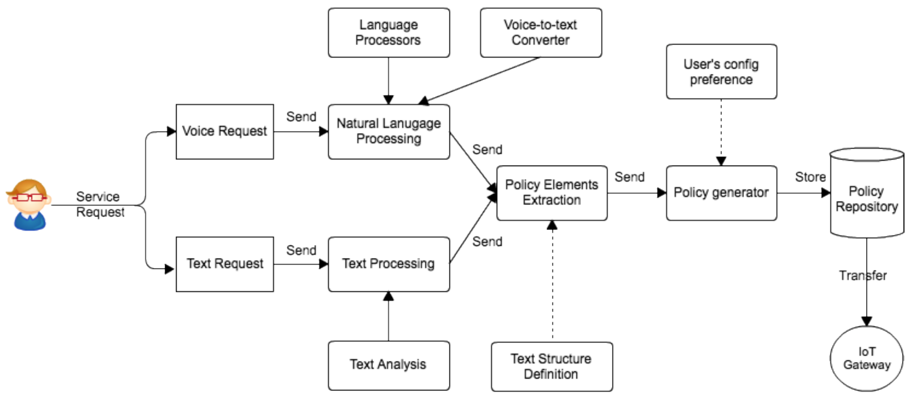


Fig. 3. PAP functional architecture

3.2 PAP System architecture

As shown in Figure 3, PAP system architecture consists of different components.

– Voice and text request. In this architecture, user can request to create polices by two different options. Either to write a text sentence or voice. Voice and text request modules are responsible for capturing user's request in voice and text and forwards user's request to further modules for analysis.
– Text processing. If user's request is text-based, the request is forwarded to "text processing" module.

- Natural Language Processing. This module is responsible for processing the recorded user's voice. This module will convert voice request to text in the pre-defined structure where information extraction can be performed by the later module.
- Language processor. Since different language may be used, it is imperative to have language processor module, which is responsible for analysing voice for different languages. For example, English, French or other languages.
- Voice to text converter. This module is responsible for converting voice request to text.
- Text analysis. It stores text mining method used to analyse the text request.
- Policy element extraction. This module is responsible for extracting the elements required to be expressed in access control policy. For example, extracting information concerning "subject", "action", "device or resource", or "condition".
- Text structure definition. This module provides the defined structure of request. It contains the official or key word used to identify different elements of policy. For example, element represent "subject".
- Policy generator. It is responsible for generating access control policy according to user's preference setting.
- User's configuration preference. This module provide the necessary system setting defined by user. It consists of access control model that user wants to use and policy languages user prefers.
- Policy repository. Once access control policy is created, it is stored at repository.

3.3 PAP model

As shown in Figure 4, the PAP model consists of the following entities.

1. Users is an entity representing the physical person or application who has rights to access PAP application. It is worth noting that we exclude the user authentication from PAP module. We suppose that authentication is a separate module from core PAP. Users entity consists of the following attributes.
   - user_id is a unique identification of user.

     user_id has the string data type.

   - user_name is a unique username, which is related to user's account.

     user_name has the string data type.

   - user_role is a profession of user.

     user_role has the string data type.

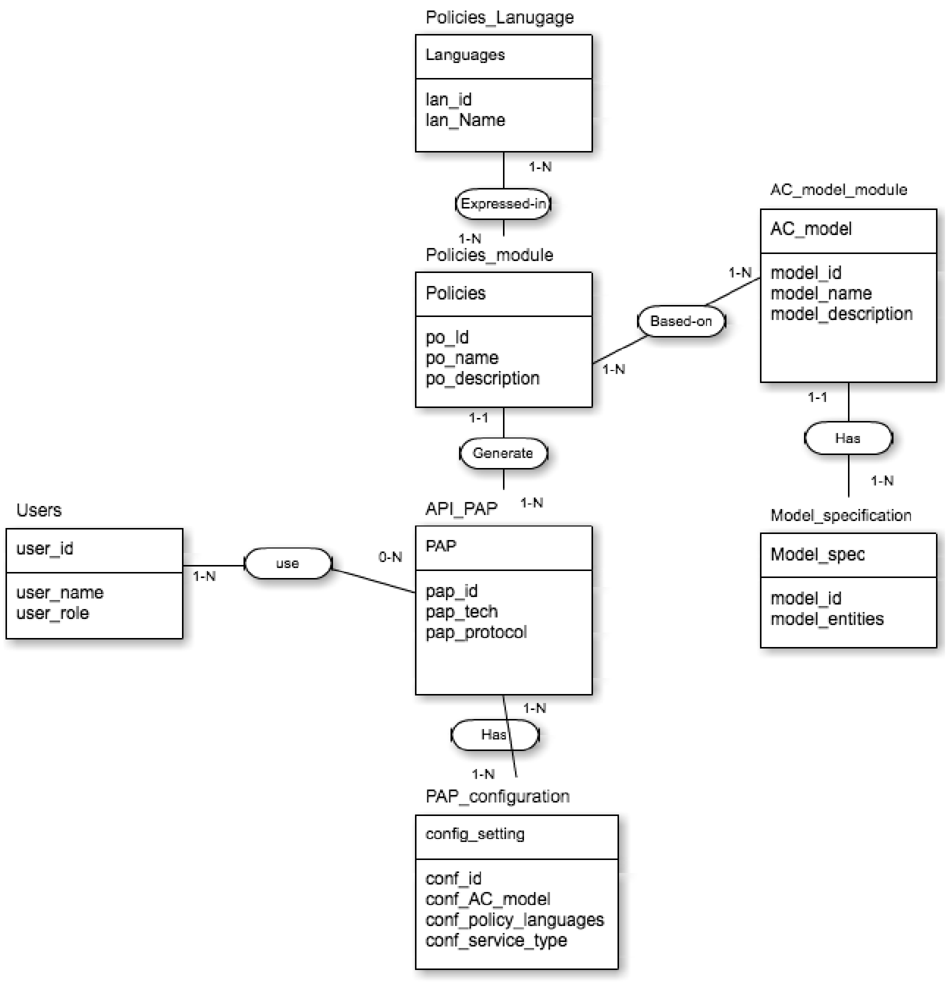


Fig. 4. PAP Entity model

2. API_PAP is the main entry for policy administration point. It associates with other entities including the PAP configuration setting and policy creation entities. PAP entity consists of the following attributes.
   - pap_id is a unique identification of PAP system. This unique identity is generated when the PAP system is installed.

     pap_id has the string data type.

   - pap_tech attribute stores the information concerning the technologies used for PAP application including the programming language, database management system, other required software and their associated versions.

     pap_tech has the string data type.

   - pap_protocol attribute stores the information concerning the communication protocols supported by PAP. This communication protocol is used for transferring the policies to external module or system.

     pap_protocol has the string data type.

3. PAP_Configuration entity store the information for API_PAP setting. Since our aim is to create PAP system that supports multiple policy languages and access control models, this configuration setting entity provides necessary information for setting API_PAP in accordance with user's preference. This entity consists of the following attributes.
   - conf_id is unique identification for a configuration setting performed by user. Each user can have his own setting.

     conf_id has the string data type.

   - conf_AC_model attribute stores information concerning the access control model that user wants to use. User can select different models. The access control policy is created based on the AC model user chooses.

     conf_AC_model has the string data type.

   - conf_policy_languages attribute stores information concerning the access control policy language that user wants to use. Once access control policy language is chosen, the system will express the access control policy in that language.

     conf_policy_languages has the string data type.

   - conf_service_type attribute stores information concerning the service PAP module provides. For example, whether PAP is accessible by other modules or not (standalone or client-server).

conf_service_type has the string data type.

4. Policies_module. This entity stores the general information concerning policies including the policy identification, policy name and the description of policy. This entity is associated with two other entities: Policies_language and AC_model_module. The Policies_module entity consists of the following attributes.
   - po_id stores unique identification of policy.

     po_id has the string data type.

   - po_name stores the policy's name.

     po_name has the string data type.

   - po_description stores the detailed description of policy.

     po_description has the string data type.

5. Policies_language entity stores information concerning the access control policy language used. This entity consists of two attributes.
   - lan_id stores the unique identification of access control policy language.

     lan_id has the string data type.

   - lan_name stores the name of access control policy language. For example, XACML (eXtensible Access Control Markup Language).

     lan_name has the string data type.

6. AC_model_module entity store the information concerning the access control module used for creating access control policies. This entity consists of the following attributes.
   - model_id stores unique identification of AC model used in the PAP.

     model_id has the string data type.

   - model_name stores the name of access control model.

     model_name has the string data type.

   - model_description stores the detailed description of AC model.

     model_description has the string data type.

7. Model_specification entity stores the detailed information concerning the selected access control model.
   - model_id stores the unique identification of model specification.

     model_id has the string data type.

   - model_entities stores the information concerning the different elements of access control model.

     model_entities has the string data type.

4 PAP communication message format

In this section, we define the PAP communication messages between PAP server and PAP client and between PAP server and other external system. This includes the policy creation request message structure, the policy creation response message structure, the policy deletion and modification message structure and the communication message structure between PAP and external system.

4.1 Policy creation request structure model

User can request to create policy by two different methods. Request to create policy by text or voice command. Once the policy request is created, it needs to be sent to PAP for generating access control policy. When the request arrives at PAP, PAP will extract the policy's elements from the request's content and generates policy according to the user's preferred setting.

Text request Text request refers to a request formed after user performed some manual selection using PAP client interface. The request format and its elements are presented below.

PRequest user CAN action resource IF conditions policy_request_id for permission and
PRequest user CANNT action resource IF conditions policy_request_id for prohibition.

Each element in policy request structure is expressed as follows.

- "PRequest" is a key word indicating the policy creation request.
- user: name_of_user (data type)
- action: name_of_action (data type)
- resource: name_of_resource (data type)
- CAN expresses the permission while CANNT signifies the prohibition.

- IF is a key word expressing conditions. Anything after IF is considered as conditions. However, the last element in the request is the policy_request_id. In general, policy_request_id is automatically generated by system.

condition structure:

- attribute_name1(data type)-operation-attribute_value
- operation:name_of_logical_operation
- attribute_name2(data type)-operation-attribute_value

The logical operations "and" and "or" are used to join multiple conditions in policy. For example, if the policy states that " Dara can turn-on TV at 6PM". Suppose that "time" is an attribute referring to time. Thus, we can write the request as following.

user: Dara (string) action:turn-on (string) resource:TV (string) conditions: time(string)-=-6PM

Voice request User can also instruct system to create policy by voice command. This voice request format is important for facilitating the speech recognition process. We define the following voice request format.

PRequest User CAN action resource IF conditions policy_request_id for permission

PRequest User CANNT action resource IF conditions policy_request_id for prohibition

- "PRequest" is a key word indicating the policy creation request.
- "User" refers to subject who has rights to perform action.
- "CAN" refers to permission while CANNT refers to prohibition.
- "action" refers to permitted action on resource. For example, turn-on, turn-off, etc.
- "resource" refers to the object or device that user wants to access.
- "IF" refers to conditions on access. Any attributes after IF are considered as conditions. However, the last element in the request is the policy_request_id.
- "Conditions structure": attribute1 operation1 attribute1_value logical_operation attribute2 operation2 attribute2_value.
- "Logical operation": "and" and "or".

4.2 Policy creation response structure model

Once PAP server receives the policy creation request by PAP client, PAP executes request and then replies back to client PAP. The response message structure is as follows.

PResponse response_id request_id policy_id user_id response_code

- "PResponse" is a key word indicating the response message.
- "response_id" is a unique response identification, which is generally automatically generated by system.
- "request_id" is a unique identification of request.
- "policy_id" is the policy identification sent in the request.
- "user_id" is a unique identification of user who requested for policy creation.
- "response_code" is the response code. 00000 response code indicating that policy cannot be created while 11111 indicates that policy is successfully created.

4.3 Policy deletion request structure model

In order to delete a policy, user needs to send policy deletion request through PAP client to PAP server. Once PAP server receives the request, it starts the process of deleting policy from policy storage. It is important to note that in order to ensure proper security protection, the deletion of policy should be allowed only to the owner of policy (policy creator) or person who has been delegated the rights by the owner. The structure of the policy deletion request format is the same for voice request or text request.

DDelete policy_id user_id request_id

- "DDelete" is a key word expressing the delete action.
- "policy_id" is unique identification of a policy. It is worth noting that for facilitating the speech recognition process, it is recommended to use number in case of deletion request by voice command.
- "user_id" is a unique identification of user who is the policy owner or person delegated by policy owner.
- "request_id" is a unique identification request. It is worth noting that this request_id is actually generated automatically by system and user can omit the request_id when he forms request command either by voice or text.

4.4 Policy deletion response structure model

After receiving deletion request from PAP client, PAP server examines the request. The policy deletion response have two possible reply codes: 0000 and 1111. 0000 indicates that the request is unsuccessful and 1111 indicates the request is successful. The response structure is as follows.

DResponse response_id request_id policy_id user_id response_code

– "DResponse" is a key word indicating the response message.
– "response_id" is a unique response identification, which is generally automatically generated by system.
– "request_id" is a unique identification of request.
– "policy_id" is the policy identification sent in the request.
– " user_id" is a unique identification of user who requested for policy deletion.
– " response_code" is the response code.

4.5 Policy modification request structure model

For policy modification, the process is more complicated because user needs to save new updated policy to policy storage. There are two different processes for policy modification. First process is the user's request for modification. The modification request is sent to PAP server. If modification is allowed, PAP server loads the policy's content and sends that contents along the policy modification response message. The second process takes place after user modified the policy. The updated policy needs to be sent to PAP server. The update request with updated policy's content is sent to PAP. Finally, PAP server executes request and sends a reply to user through PAP client.

1. **Policy modification request** has the following structure. Each element is separated by space.

   MRequest policy_id user_id request_id

   – "MRequest" is a key word indicating the policy modification request.
   – "policy_id" is a unique identification of policy.
   – "user_id" is a unique user identification.
   – "request_id" is a request identification, which is automatically generated by system.
2. **Policy modification response.** After receiving the policy modification request, PAP server executes request and responds to the request with the following structure.

   MResponse policy_id user_id request_id policy's content response_code

   – "MResponse" is a key word indicating the response message of policy modification request.
   – "policy's content" is a policy's content. The content consists of the following elements: user, action, resource and conditions.
   – "response_code" is code generated by PAP server. There are two different response codes. 000000 indicates the the modification request is unsuccessful and 111111 indicates otherwise. It is worth noting that if response code is 000000, policy's content can be empty.

3. if the response is positive, PAP client extract policy's content from the response message and make them available for user to modify them. Once user finishes the modification, PAP client sends policy update request to PAP server and PAP server returns update response back to PAP client. The policy update request has the following structure.

   URequest policy_id user_id request_id

4. The return updated response from PAP server has the following structure.

   UResponse policy_id user_id request_id response_code

   – "UResponse" is a key word indicating the update response message.
   – "response_code" is a code indicating the different states of update request execution performed by PAP server. There are two different codes: 0000000 for unsuccessful execution and 1111111 for successful execution.

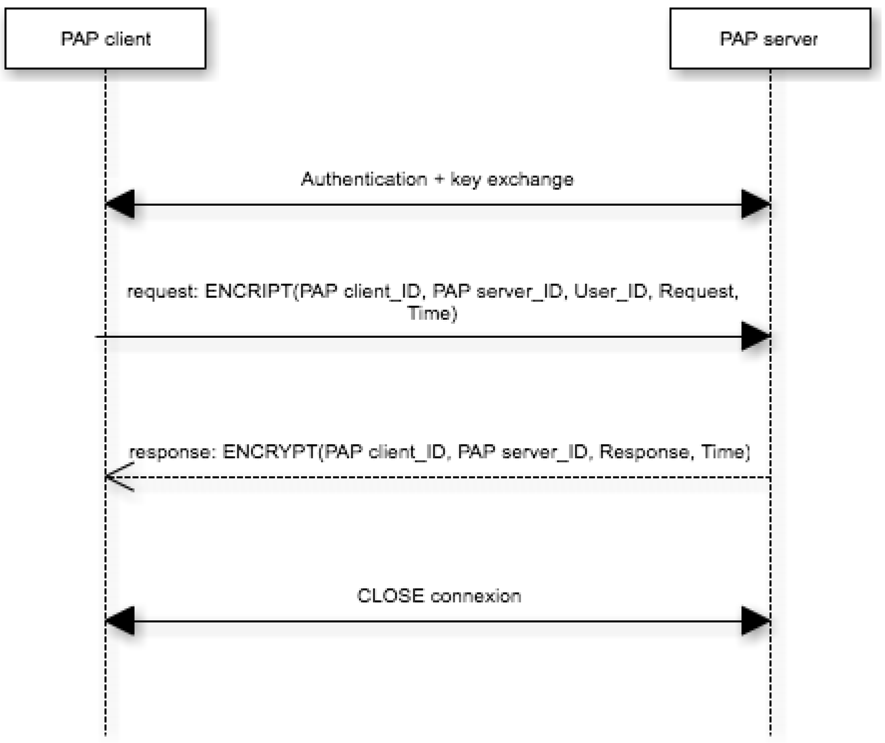

PAP client
PAP server
Authentication + key exchange
request: ENCRIPT(PAP client_ID, PAP server_ID, User_ID, Request, Time)
response: ENCRYPT(PAP client_ID, PAP server_ID, Response, Time)
CLOSE connexion

Fig. 5. Communication protocol between PAP client and PAP server

4.6 Communication Protocol between PAP client and PAP Server

In general, user uses PAP client application for managing access control policy. However, most of the execution processes take place in PAP server. PAP client needs to communicate with PAP server and the messages exchanged between the two entities must happen in the secure manner. Thus, we propose the communication protocol between PAP client and PAP server as shown in Figure 5, the sequence diagram showing different communication phases. Each communication between PAP client and PAP server is separated into 3 main phases.

1. Authentication phase. Both PAP client and server need to prove their identity to each other and exchange the key for using in next step communication.
2. Message exchange phase. Once the authentication is done, client needs to send the request to server for further execution. There are different types of request, such as request to create policy, modify policy and delete policy. Every request message is encrypted with symmetric key (key that the PAP client and server exchanged during the authentication phase) and the message contain the following elements.
   – PAP client_ID is a unique identification of PAP client application.
   – PAP server_ID is a unique identification of PAP server.
   – User_ID is a unique identification of user who is currently using PAP application and instructing the request.
   – Request is a package of data containing different data elements depending on different types of request (see Section IV.A to F).
   – Time is a time at which the request is initiated.

   After executing request from PAP client, PAP server responds and the response message has the following structure.
   – PAP client_ID is a unique identification of PAP client application.
   – PAP server_ID is a unique identification of PAP server.
   – Response contains the data elements depending on different types of request (see Section IV.A to F).
   – Time is a time at which response message is sent to PAP client.

   It is worth noting that response message needs also to be encrypted and the encryption key is from the key exchanged during the authentication phase.
3. Once the communication between PAP client and server is finished, the connexion is closed and the key is also destroyed.

4.7 Communication Protocol between PAP server and IoT hub

In the scenario where PAP server needs to upload access control policies to IoT hub where policy decision takes place, a secure communication between the two entities should be assured. Figure 6 provides the communication protocol for exchanging message between PAP server and IoT hub. As shown in Figure 6, there are three communication phases.

1. Authentication phase. In this phase, both PAP server and IoT hub need to prove themselves to each other. They also exchange the symmetric key for securing further communication between them.

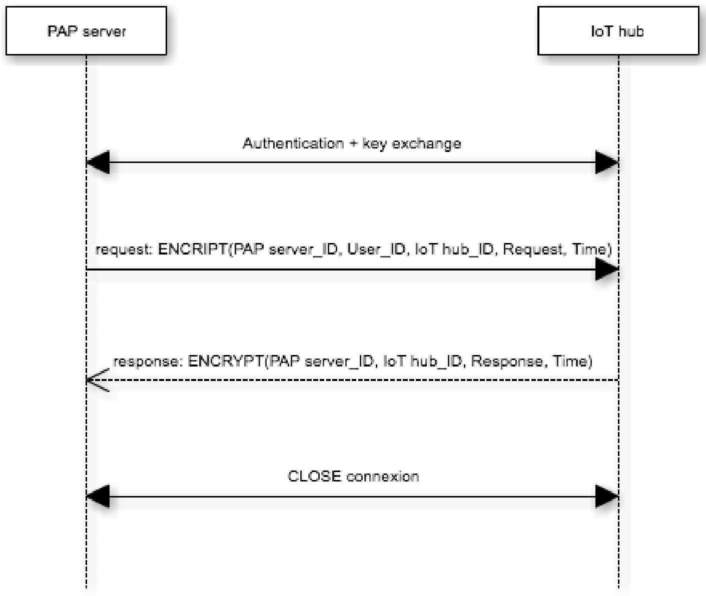

PAP server
IoT hub
Authentication + key exchange
request: ENCRIPT(PAP server_ID, User_ID, IoT hub_ID, Request, Time)
response: ENCRYPT(PAP server_ID, IoT hub_ID, Response, Time)
CLOSE connexion

Fig. 6. Communication protocol between PAP server and IoT hub

2. Once the authentication is finished, PAP server starts the policy uploading process. PAP server sends the policy uploading request, which contains the following data elements.
   - PAP server_ID is a unique identification of the PAP server in the IoT system.
   - User_ID is a unique identity of user who manages PAP server.
   - IoT hub_ID is a unique identity of IoT hub.
   - Request contains the request data elements. For example, in case of uploading policy, the policy's content is the request's content.
   - Time is a time at which the upload process is initiated.

   After receiving request from PAP server, IoT hub starts to process the request and once the process is finished, IoT hub responds to PAP server. The response structure consists of the following elements.
   - PAP server_ID is a unique identification of the PAP server in the IoT system.
   - User_ID is a unique identity of user who manages PAP server.
   - IoT hub_ID is a unique identity of IoT hub.
   - Response contains the response data elements and the response code. There are two types of response code: 000000000 for failed request and 111111111 for successful request.
   - Time is a time at which the upload process is initiated.
3. Once the communication between PAP client and server is finished, the connexion is closed and the key is also destroyed.

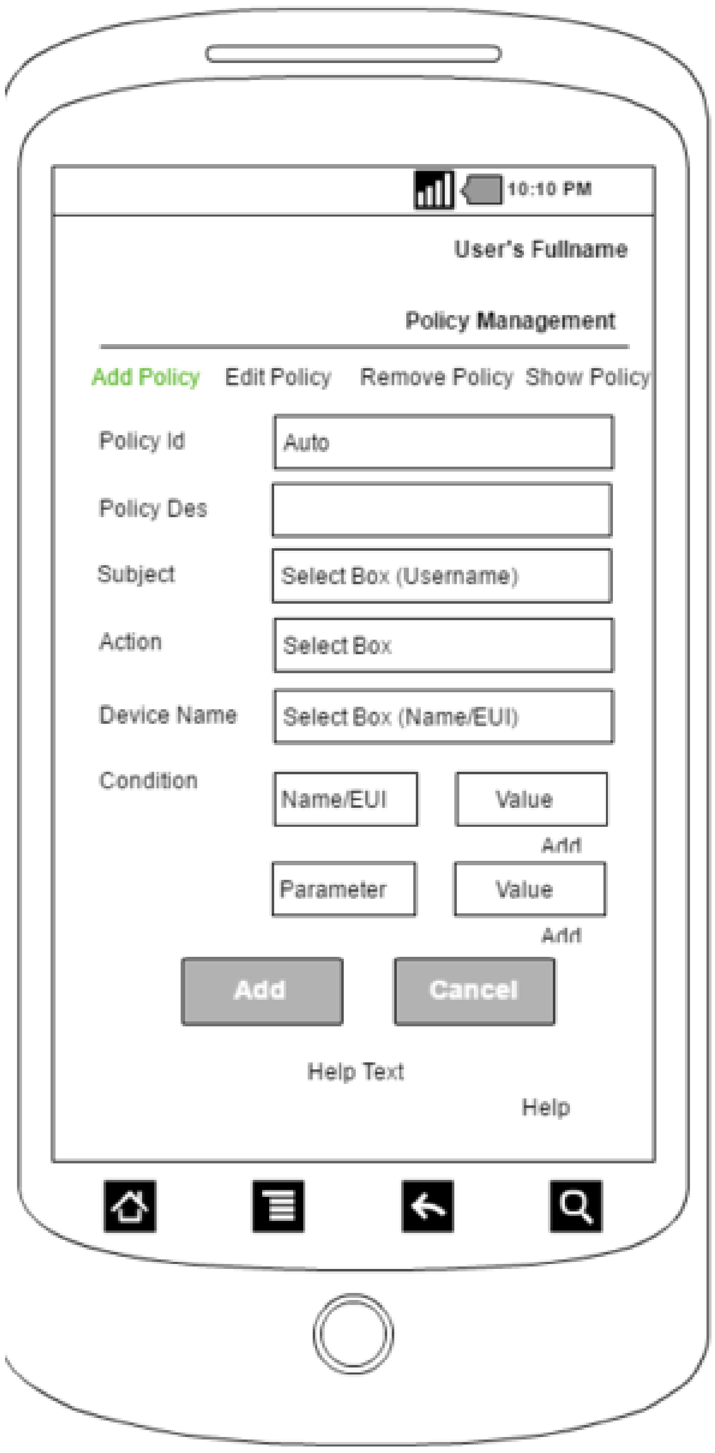


Fig. 7. Add policy interface

5 PAP Interface design

In this section, we focus on the implementation of the PAP Server, PAP Client and the communication protocol between PAP client and PAP server and PAP server and IoT hub.

5.1 PAP Client Interface Design

PAP client interface allows user to connect to PAP server with his preferred setting. PAP client interface allows user to also choose different access control models and policy languages that user wants to use. It also allows user to set different communication protocols between PAP client and PAP server. The PAP client interface is divided into two main parts: policy management part and application setting part. Policy management section consists of the tools used to create, modify, delete and show policy. The setting section dedicates to PAP client setting. That includes the communication protocol, access control model, policy language, IoT devices setting and user setting.

Policy management This part consists of four management tools.

– Add policy. Figure 7 shows the proposed designed interface for add policy tool. When creating policy some data elements need to be filled, such as the unique policy identification, the detailed policy description, the user (or subject) who is allowed to access data or device, the action on device, the device name or identification and the conditions. It is worth noting that different conditions can also be joined by logical operations, such as “and” or “or”.
– Edit or modify policy. In some cases, user may want to edit or modify the policy he created. This edit policy option is created for that purpose. In this designed interface (see Figure 8), when user wants to edit policy, he can select the policy_id from policy_id select box and once policy is selected, the policy’s content is loaded and filled in the corresponding text fields automatically. The data elements in edit option interface is the same as that of add option interface.
– Show policy. Figure 9 is the PAP client interface for show policy option. When user wants to show the policy’s content for the purpose of examining it, user can do so by selecting policy from the select box. In that select box there are two information that are necessary for recognising which policy is for which user and device. The information in the select box consists of policy_id and the policy description.
– Remove policy. Figure 10 shows the design of policy deletion interface where user can remove policy from policy storage. When user wants to delete policy from storage, he can simply select the policy id and click on remove button. The PAP client interface will alert message once gain to confirm the policy deletion instruction. If user confirms his request, the policy deletion instruction is sent to PAP server for execution.

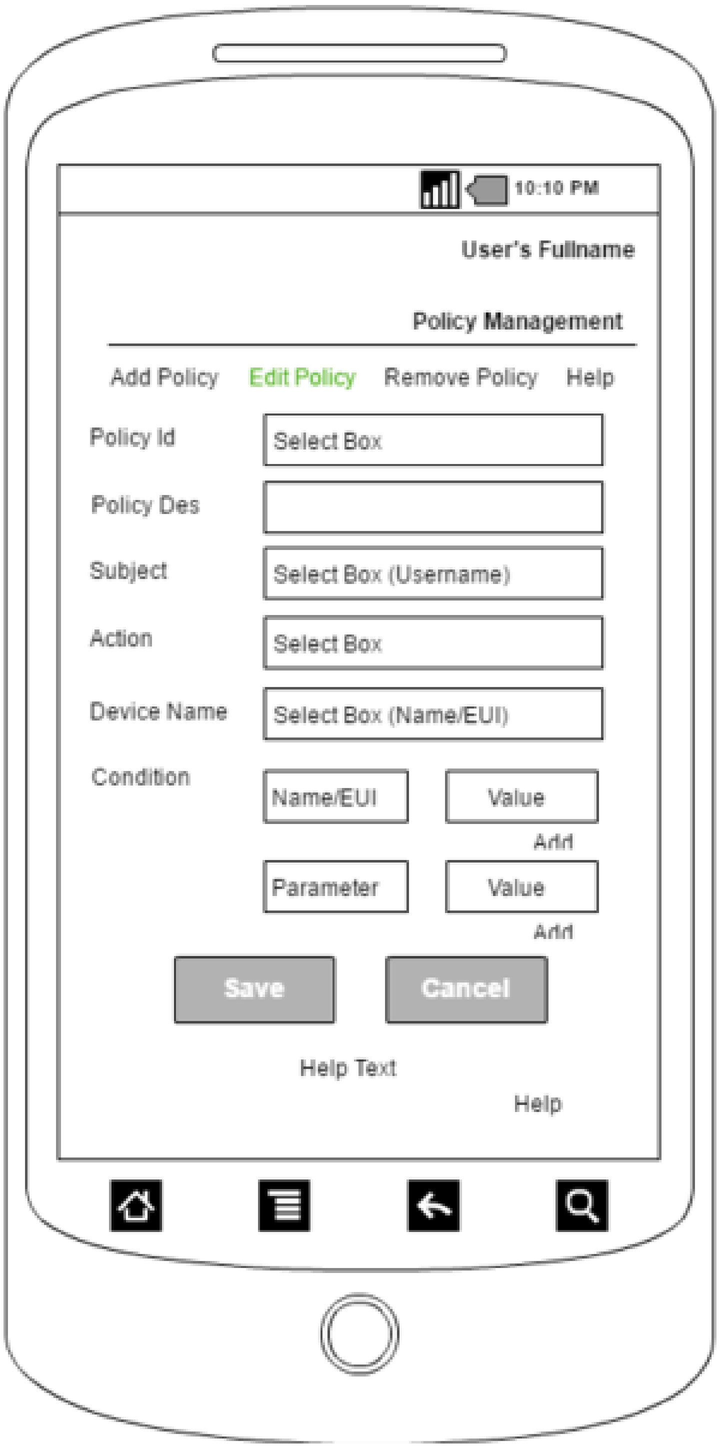


Fig. 8. Edit policy interface

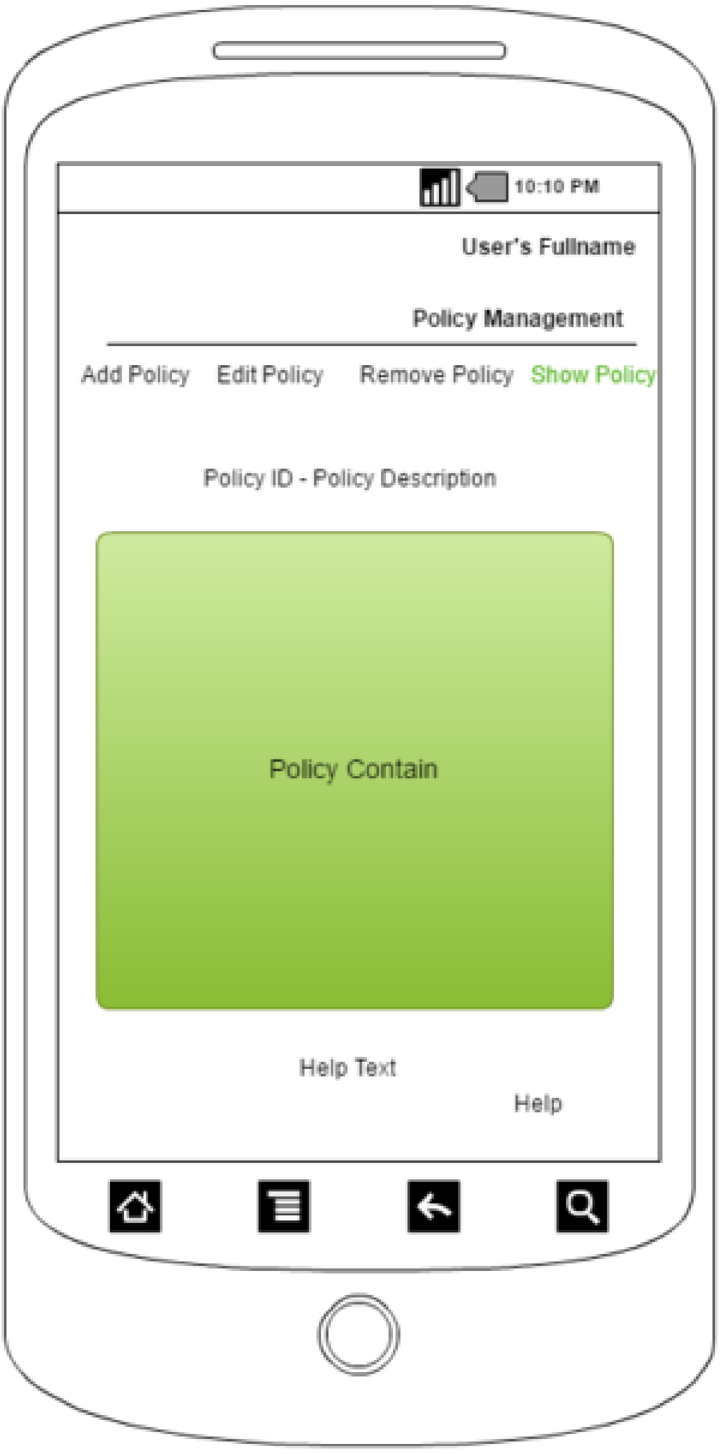


Fig. 9. Show policy interface

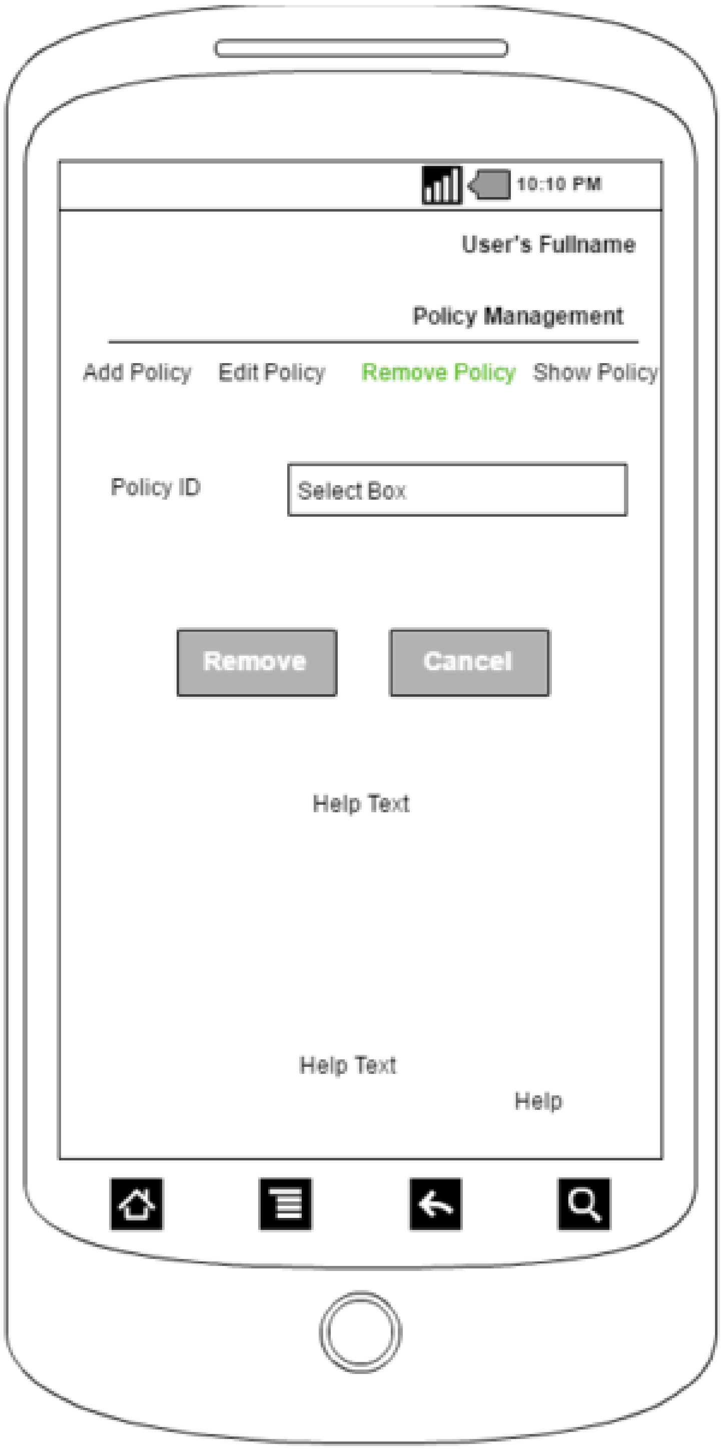
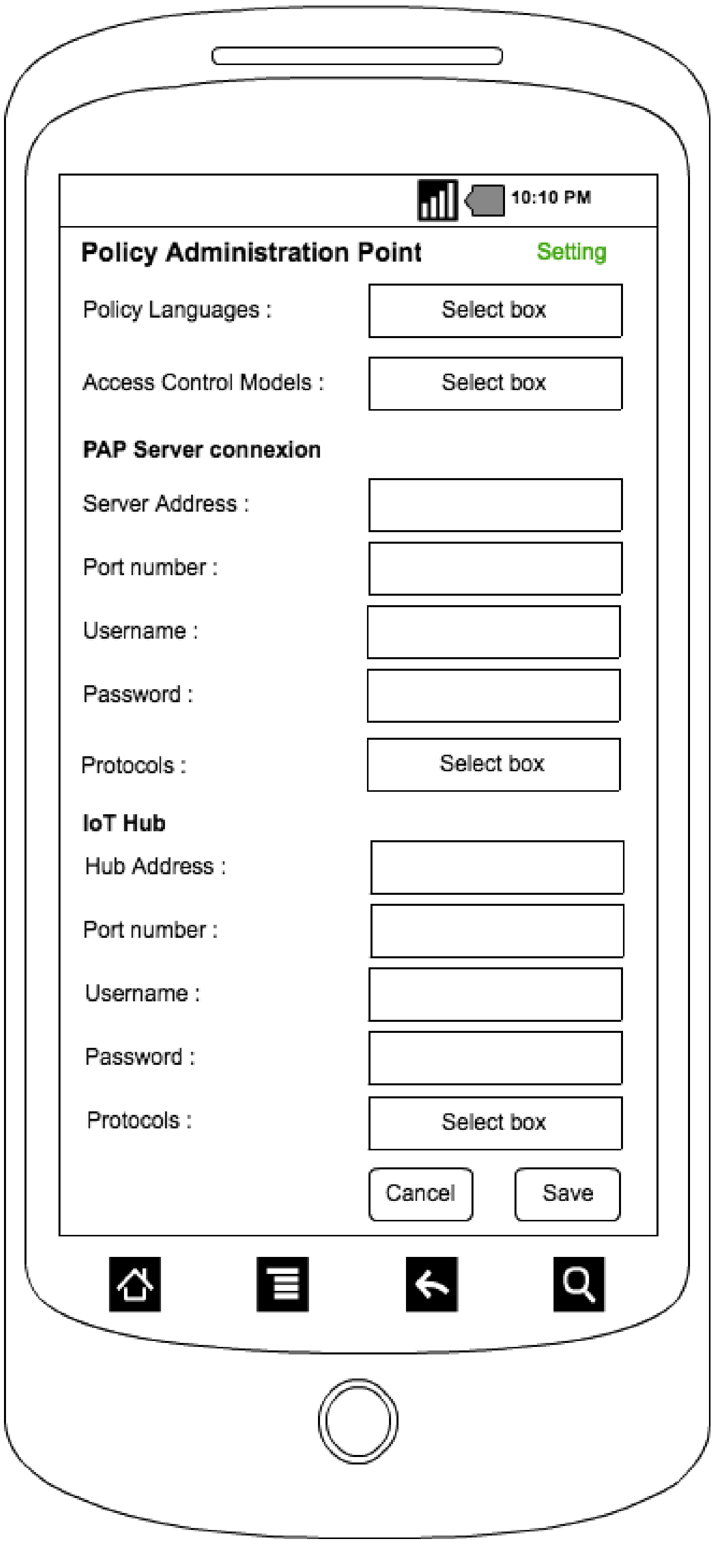


Fig. 10. Remove policy interface


10:10 PM

Policy Administration Point Setting

Policy Languages :	Select box
Access Control Models :	Select box

PAP Server connexion

Server Address :	
Port number :	
Username :	
Password :	
Protocols :	Select box

IoT Hub

Hub Address :	
Port number :	
Username :	
Password :	
Protocols :	Select box

Cancel Save

Fig. 11. PAP setting

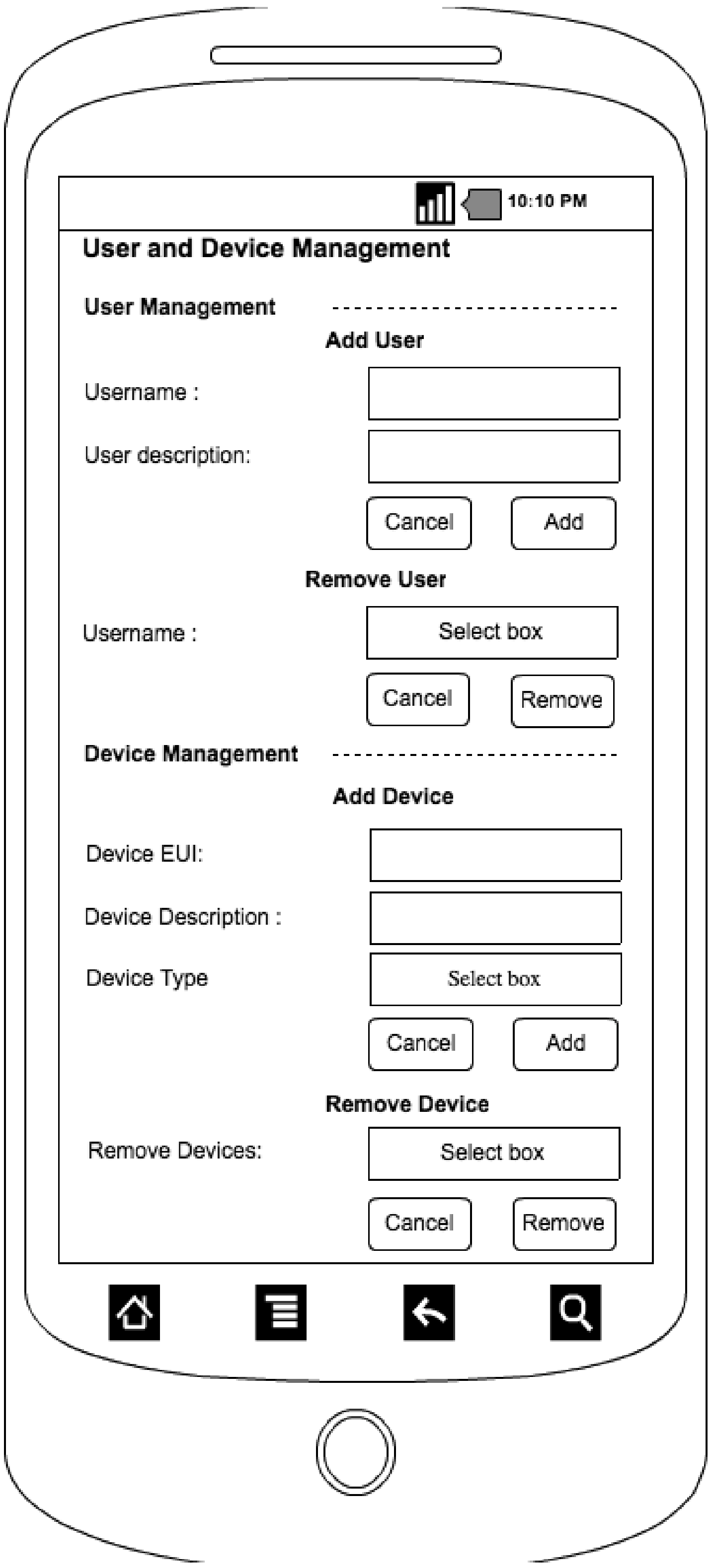


Fig. 12. User and Device management

PAP client setting there are two different settings for PAP clients. The first setting is for connecting PAP clients to PAP server and the IoT hub if exists. The second setting relates to user and IoT devices management. The second setting allows PAP administrator to add users who may later be granted to access devices. PAP administrator can also add devices to IoT system through this setting.

1. Policy administration point setting. There are three main sections as shown in Figure 11.
   - The first section, user can set policy language and access control model he prefers. Once the policy language and access control model are selected, PAP client will self configure and generate access control policy based on the selected access control model and the policy language user chose.
   - The second section relates to the PAP server connexion setting. This section consists of different attributes that user needs to fill up.
     - Server address is the IP address of the PAP server where the PAP client must connect to.
     - Port number is the PAP server communication port number.
     - Username is a name of user account that user uses to connect to server.
     - Password. If password is used, user must also set password to be able to connect to PAP server.
     - Protocols. User must also select the communication protocol between PAP client and PAP server. In case different protocols are used, the chosen one is used as the default communication protocol.
   - The third section relates to IoT hub configuration. In the IoT scenario in Figure 2, PAP server needs to upload access control policy, after creating it, to IoT hub. Thus, it is important to set IoT hub information with which PAP server can use to self configure to connect to IoT hub.
     - IoT hub IP address is the IP address of IoT hub in the network. It is worth noting that although there may be more than one hubs in IoT system, PAP client application can only connect to one hub at a time.
     - Port number is a communication port number of the server running in the IoT hub.
     - Username is the name of user account created in the IoT hub server.
     - Password is the password of the user account created in the IoT hub server.
     - Protocols. User must also select the communication protocol between PAP server and IoT hub. In case different protocols are used, the chosen one is used as the default communication protocol.
2. User and device management. This setting is designed to allow user to be able to add devices and users and also to remove them if they are not in use.
   - User management. PAP client administrator can add user to the system. Two information attributes need to be filled. The username of user and the general description. The user description provides the detailed

information of user in case username can not be used. In case PAP administrator wants to remove user from the system, he can do it by selecting username from the select box and then clicks on the remove button. It is worth noting that if the user is removed, the access control policy concerning this user needs also to be removed from the AC policy storage. In our implementation, it will not be automatically remove, PAP client administrator needs to remove AC policy by using AC policy management tool.

– Device management allows PAP client administrator to add new device to the IoT system and also removes existing devices from system.
  - Device EUI is a unique identification of IoT device.
  - Device Description is a general description of the device, such as sensor or actuator.
  - Device type is the type of data that device generates. For example, temperature sensor generates float data type and so on.

6 Conclusion

In this paper, we present the first draft of the design of PAP client application for smart home system environment. The presentation includes: the global architecture of the smart home system with PAP integration, the functional requirements and global architecture of PAP, PAP communication message structure and format and the PAP client interface design. Our next step is to design the detailed PAP client implementation for the proof-of-concept and the work on access control model and access control policy languages. We also envisage to work on different communication security protocols and encryption methods to be used in our implementation.

References

1. Bruce Ndibanje, Hoon-Jae Lee and Sang-Gon Lee. Security Analysis and Improvement of Authentication and Access Control in the Internet of Things. Open access Sensors, 14(8), 14786-14805; doi:10.3390/s140814786, 2014.
2. Rahul Godha, Sneh Prateek and Nikhita Kataria. Home Au- tomation: Access Control for IoT Devices. International Journal of Scientific and Research Publication, Volume 4, Issue 10, October 2014, ISSN 2250-3153.
3. Blase Ur, Jaeyeon Jung and Stuart Schechter. The current State of Access Control for Smart Devices in Homes. Workshop on Home Usable Privacy and Security (HUPS), July 24-26, 2013, Newcastle, UK.
4. Ricardo Neisse, Gary Steri, Igor Nai Fovino and Gianmarco Baldini. SecKit: A Model-based Security Toolkit for the Internet of Things. The journal of Computer and Security (2015), page 60-76. Published by ELSEVIER.
5. Sachin Babar, Parikshit Mahalle, Antonietta Stango, Neeli Prasad and Ramjee Prasad. Proposed Security Model and Threat Taxonomy for the Internet of Things. International Conference on Network Security and Applications (CNSA 2010). Recent Trends in Network Security and Applications pp 420-429. Published in Springer 2010.

6. J. Sathish Kular and Dhiren R. Patel. A survey on Internet of Things: Security and Privacy Issues. International Journal of Computer Applications. Volume 90. No 11, March 2014.
7. Assesement of Access Control Systems. National Institute of Standards and Technology. Technology Administration U.S. Department of Commerce. http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf
8. Smart Home service providers. http://www.sensorsmag.com/components/top-10-smart- home-service-providers-us
9. Definition of cloud service. http://www.webopedia.com/TERM/C/cloud services.html
10. Agrawal, Rakesh and Imielin?ski, Tomasz and Swami, Arun. Mining Association Rules Between Sets of Items in Large Databases. SIGMOD Rec. June 1, 1993. Vol 22, No 2., New York, NY, USA.
11. Extensible Markup Language (XACML). http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
12. M. Schiefer. Smart Home Definition and Security Threats. Ninth International Conference on IT Security Incident Man- agement IT Forensics. pages. 114-118, May 2015.
13. Jose L. Hernandez-Ramos, Antonio J. Jara, Leandro Marin and Antonio F. Skarmeta. Distributed Capability-based Access Control for the Internet of Things. Journal of Internet Services and Information Security (JISIS), volume: 3, number: 3/4, pp. 1-16.
14. N. Komninos and E. Philippou and A. Pitsillides. Survey in Smart Grid and Smart Home Security: Issues, Challenges and Countermeasures. IEEE Communications Surveys Tutorials. vol. 16, No. 4, P. 1933-1954, 2014.
15. Wireless Technologies. https://www.link-labs.com/blog/types- of-wireless-technology
16. Samsung smart thing . https://www.smartthings.com/
17. One M2M specification. http://www.onem2m.org/
18. Logitech harmony. http://www.logitech.com/en- us/product/harmony-hub
19. Google hub. https://on.google.com/hub/