Deliverable 3.3.1

Project Title	Next-Generation Hybrid Broadcast Broadband
Project Acronym	HBB-NEXT
Call Identifier	FP7-ICT-2011-7
Starting Date	01.10.2011
End Date	31.03.2014
Contract no.	287848
Deliverable no.	3.3.1
Deliverable Name	Design and Protocol: Intermediate User ID, Profile, Application Reputation Framework
Work package	3
Nature	Report
Dissemination	Public
Authors	Gregor Rozinaj, Ivan Minárik, Jarmila Pavlovičová (STUBA), Félix Gómez Mármol, Ginés Dólera Tormo (NEC), Mark Gülbahar (IRT), Sebastian Schumann (ST)
Contributors	Juraj Kačur, Matej Féder, Jozef Bán, Ľuboš Omelina, Miloš Oravec, Marek Vančo, Alexandra Posoldová (STUBA), Ondrej Lábaj (ST)
Due Date	31.05.13
Actual Delivery Date	31.05.13

Table of Contents

Executive Summary

This document is the third deliverable of WP3 of the HBB-NEXT project and constitutes Milestone 7 for the WP3 enablers: “MS7: The intermediate version of the software components of WP3, 4 and 5 is in place”.

It presents designs, first prototype implementations and the basic integration of modules related to personalisation of next-generation hybrid-broadcast-broadband television. It provides a current (May 2013) view of the status of the designs, prototypes, integrations and demonstrations towards a central use case. It also includes an outlook to future work.

Use case: The principal use case of WP3 is described in following scenario.

- Generic app store
- User is entering the room, basic identification recognizes him/her
- Personalized app store will appear
- User tries to install/buy/validate(feedback) the app
- According user multilevel authentication and behaviour user is allowed/not allowed to do that

Design: A preliminary system design is presented, based on the following components:

- Multimodal interface
- Identity and Security Manager
- Reputation Framework
- AppStore

Prototypes: The following initial prototypes have been realised: Face Recognition (STUBA), Gesture Recognition (STUBA), Speaker identification (STUBA), Integration System (STUBA), Application reputation system (NEC) and AppStore (NEC). The first four prototypes are a part of multimodal interface made by STUBA while the last application from NEC represents reputation framework.

Integration: Within WP3 we have presented the first successful integration of the two following components: gesture recognition and remote control of the TV system using gestures. Furthermore, we continue on the integration of the whole system according the scenario.

Demonstration: Demonstration of face recognition, speaker identification, as well as reputation framework has been presented. Demonstration of the basic AppStore functionality has been prepared, as well.

Future work: Among other the following work is planned for WP3 for the next year: improving the multi-modal interface, combining face recognition with voice recognition and enabling the identification of multilevel user identification; 3D face recognition and iris recognition. Improving of the reputation framework should result in a possible integration of all WP3 functionalities into one integrated demo.

1. Introduction

1.1. Role in the System

One of the major objectives of HBB-NEXT is to define an open framework for the next generation of hybrid broadcast internet (HBI or HBB) services. Building on top of deployed standards like HbbTV, HBB-NEXT defines a set of extensions and open APIs to enable more advanced services and business cases. On business models please refer to D2.4 [17] for more details.

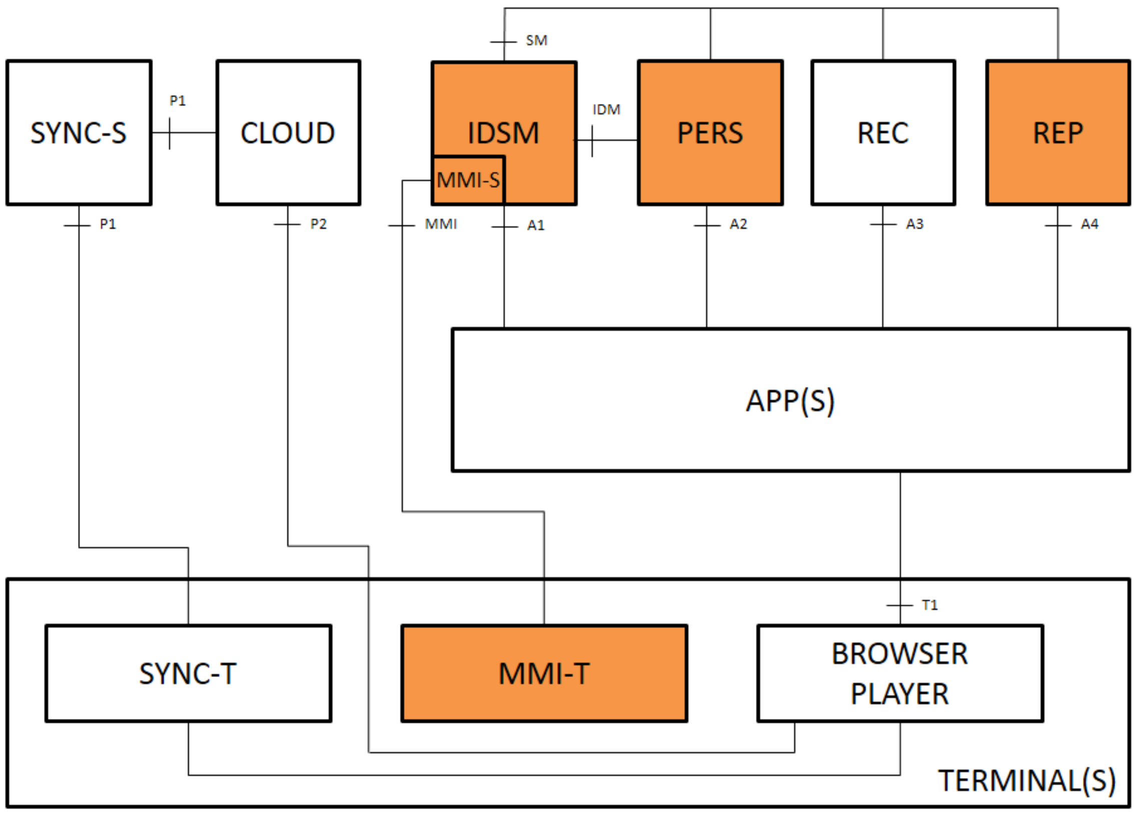


Figure 1: WP3 modules in the HBB-NEXT system architecture

Figure 1 shows the system architecture of the HBB-NEXT platform as developed in WP6. The diagram consists of enablers provided by WP3, which either reside on the internet (cloud) or on the terminal, and public interfaces to the HBB-NEXT applications (APPS). Interfaces in the diagram can be distinguished into APIs and protocols. APIs of internet based enablers are potentially exposed by one or multiple service provider and named A1, A2 etc. The API that is exposed to applications on the terminal is named T1.

HBB-NEXT also defines open protocols for the advanced synchronization features. These protocol interfaces are named P1 and P2. APIs between enablers are labelled with the name of the sub module that exposes it, MMI, IDM and SM.

WP3 contributes to the user identification on various levels of security using multimodal interface and to the Reputation Framework module.

This deliverable provides first designs and protocols for the following components:

- Multimodal User Identification
- Security Manager
- Identity Manager
- Reputation Framework.

Multimodal interface has been addressed by speaker identification and face recognition. Furthermore, 3D face recognition has been introduced to offer secure identification. The modules are designed to cooperate together and to demonstrate the integration on a simple demo. A user enters the room, the system will recognize the user, the user opens the AppStore application and the system allows him to choose, open, buy and install a desired application. For each activity or operation of the user, the system may ask multilevel authentication based on secure identification with satisfactory validation and security.

The structure of this document is as follows. The second chapter is devoted to multimodal interface and research areas, in which the project team implemented applications for user identification. The third chapter describes trust and reputation framework. Identity management module, Security management module and Profile management module described in following 3 chapters together with multimodal interface module, offer the background for secure and personalised functionality of the system.

2. Multimodal User Identification / Authentication, Multilevel Authorization

Multimodal User Identification is one of the key features of the HBB-NEXT project aiming at effortless utilization of the system's features while assuring that the system only performs commands which are properly authorized. The speaker identification tends to provide basic identification of the possible users located in the system installation area. The basic level would be suitable for not-so-crucial identification tasks, such as loading personal profile. The face detection approach aims to provide more reliable user identification based on users' faces which contain far more characteristics that can be parameterised in comparison to the voice identification approach. Additionally, the 3D face recognition further extends the possibilities of feature extraction in order to more precisely identify particular persons and can be thus used for the highest level authentication (and authorisation) for the most demanding applications (i.e. bank account login, etc.). For security reasons, 3D face recognition authentication can be enhanced with e.g. eye movement tracking. This approach may simulate multi-factor authentication (login plus token) necessary for the highest level authentication.

2.1. Speaker Identification

The speaker recognition task is getting more and more popular with the emerging advances in technology and theory. This complex task is further divided into several categories according different criteria. The aim of a speaker identification system is to detect the identity of a speaker upon an utterance, regardless of what he or she said. As any speaker identification system consists of two main parts, namely: feature extraction and classification method, the designer has to select proper methods and their modifications for a given application which may differ depending on type and setting of particular task. Detailed overview on speaker recognition can be found in [2] and [4]. For the purpose of our application we have chosen the k- nearest neighbour method (KNN) [7] for classification and Mel frequency cepstral coefficients (MFCC) [5] and their modifications for speech parameterizations. However, the model based generative method of GMM and its variations [7] are under close investigation and are being implemented into the final system as well, [3]. For the system functional overview see Figure 2 that will be explained in the Interface section.

2.1.1. Theoretical Basis- research achievements

Based on the previous theoretical description let us present the research activities and main achievements recorded so far.

In the frame of the project several research activities has been launched in order to increase the accuracy as well as the robustness of the identification system. In the first step several well-known and accepted methods for speaker identification task were selected for further experiments and optimization process, to name just few: MFCC, PLP [6], KNN, GMM, etc.

Based on that initial selection and optimization of known methods based on a newly designed and recorded training and testing database has been carried out. This process can be divided into 3 areas [4], namely: feature extraction methods, feature normalization methods and classification methods.

In the context of feature extraction methods following approaches has been investigated: MFCC, PLP, CLPC, Mel filter bank, Mel filter bank & PCA, and Mel filter bank with all pole modelling. For plain MFCC following parameters has been tested and optimally set: frequency ranges (300-9000Hz) where the lowest identification errors are observed, number of filter banks equally spaced in the Mel frequency spectra (28- this number was derived as the maximum limit based on the sampling frequency and frequency ranges), and number of cepstral coefficients (c1 to c20). Some results regarding the accuracy and free MFCC parameters are shown in Table 1.

	upper frequency limits				
lower frequency limits 175Hz	4000	6000	8000	9000	10000
averaged accuracy	78.88	88.36	93.33	94.05	93.26
minimal accuracy	6.67	40	71.43	71.43	61.54
	lower frequency limits				
upper frequency limits 9000Hz	100	175	300		
averaged accuracy	93.4	94	94.3		
minimal accuracy	71.4	71.4	76.9		

Table 1: The frequency range optimization of MFCCs for speaker recognition.

Similar experiments have been accomplished for other features; their comparison in terms of achieved accuracies is shown in Table 2.

Features	MFCC	PLP	CLPC	MEL CLPC	Mel fbank & PCA
Mean accuracy [%]	95	76.5	83.3	80	96.2
Min. accuracy [%]	80	45	41.1	47.3	61.5

Table 2: The comparison of feature extraction methods for speaker identification.

In the domain of classification 2 different approaches has been tested: KNN (sample based, non-parametric method) and GMM (parametric, generative method). For GMM method the optimal number of Gaussian mixtures and different initializations methods has been tested. It has been observed that for approx. 20 speakers of both genders the best number of mixtures using 27 MFCC features lies in the range of 16 to 32. In the context of GMM initialization 3 method has been tested, i.e.: a random distribution of centres, means, and whole models derived for a general speaker. The latter two proved to be more accurate and robust than the randomly set centres. However, in the test scenarios the best GMM based models have been outperformed by a simple sample based KNN classification. The best averaged accuracy for GMM was 93% vs. 95% in the case of KNN; in the terms of worst classifications recorded per a single user it was 70% for GMM and 80% for KNN. Thus in the following more focus has been put on the KNN method and all its variations with many free parameters to set. In the optimization process we assumed approx. 20 speakers each with 15s of eligible training speech and 3s of testing speech (trade of between accuracy and time delay). Therefore the following parameters have been subject to the optimization process: number of nearest detected neighbours (3-5), used local distances (EUCLID, EUCLID^2, absolute value, and all these also in their relative versions- related to the vectors' norms), and distance weighted KNN vs. majority rule KNN (distance weighted proved to be more robust). In addition different weighted local distances based on Euclid one have been proposed and tested (sin window, Gauss window, exponential window, windows compensating differences in dispersions or powers among features). It was found that sine windows consistently provided one of the best results reaching up to 95%.

In order to compensate inter-session variability methods like spectral subtraction, and power normalizations have been tested with different settings, however their basic forms fail to produce any improvement in real time conditions. Furthermore the well-known PCA [8] method was tested as a form of unsupervised feature normalization process but its real time application was not so beneficial either.

Unlike previously mentioned research achievements that are related mainly to finding and setting optimal values of classical methods in order to provide optimal results when cooperating in a speaker identification task, new methods, modifications to existing ones or new combinations of methods has been also designed and tested.

New voice activity detection was designed for real time processing so that it is simple and fast. It is based on a filtered energy and zero crossing statistics conditioned by some intuitive absolute threshold. It was set to eliminate the speech miss ratio.

Further, modification to the classical MFCC feature extraction method has been proposed, replacing the DCT transform with PCA [8] and LDA [9] transforms in order to provide more informative and discriminative features in a more compact form (Mel filter bank with PCA & LDA). This combination proved to be beneficial however is prone to the changes in the recording environment. Thus more environments must be tested and used in the construction of an optimal transform (optimal from a linear point of view). Selecting those distinctive environments and constructing specialized transforms is an open issue.

So called- weighted mean subtraction and weighted power/ dispersion normalization techniques have been designed and tested, as the inter session modifications may not be equally allocated among feature elements. Testing only several windows and environments, these approaches slightly outperformed the classical ones (88% vs. 76%) however they are still not beneficial for real systems, so far.

The basic KNN algorithm was extended for the purpose of a speaker identification system so that it operates on several distinctive categories (spectral) determined by VQ or GMM. For each category an unique local distance is found that more precisely match the data and by doing so even better results were observed than when using the classical concept with the best settings, either for the environmental match scenarios (96.2% vs. 95.1%) or in mismatch ones (72% vs. 56%).

To suppress the KNN inborn bias decision towards most frequent speakers present in the training database a confidence weighting algorithm was designed and implemented.

2.1.2. Data Model

In order to perform speaker identification using the selected KNN method (weighted KNN) and MFCC, following internal data structures had to be introduced:

- Buffer of speech samples, which is a simple vector containing sequence of short numbers. As there are both recognition and training phases which are of the different lengths two such buffers of different sizes are used.
- Buffer of parameterised speech, i.e. vectors of MFCC. Theoretically it is a matrix where rows are different speech frames and columns are individual features (MFCC elements). However it is implemented as a vector. 2 buffers are used one for tested speech and the second one will store all vectors in the database (training MFCC vectors appended by indexes of speakers)
- Vectors for found neighbours (their indexes)
- Vectors for found neighbours (their distances)
- Vector of winners (their indexes)
- Vector of winners (their confidences)
- Structure for storing recognition settings that contains 10 important items controlling the recognition process, like: type of local distance, umber of neighbours, training and recognition times etc.
- Structure for storing feature extraction settings that contains 12 important items controlling the speech extraction process, like: frequency limits, frame length and its shift etc.
- Array of strings storing indexes of speakers and their names

2.1.4. Design

The application is a standalone console Windows application and as it should be fast and process lot of data it is coded in C/ C++ language. As it uses OS and hardware specific system calls (reading sounds from microphones via WINAPI 32) it is not portable to other OS systems unlike Windows 98 and higher versions.

The basic facilities of the applications are:

1. Recording a new speaker, transforming the wave samples into MFCC features according to configuration variables and storing both name and feature vectors in the database
2. Identifying unknown speaker (recording voice, producing MFCC features, finding nearest neighbours and taking decision) according to the configuration variables.
3. Listing recorded users in the database.
4. Removing recorded users from the database
5. Change its functionality via properly setting decision and parameterisation configuration files.

2.1.5. Interface

Internally the application can be divided into several functionally distinctive and setting independent blocks which are: wave form reading, feature extraction (MFCC), KNN, final decision taking, I/O functions, data storage and user API. Their mutual cooperation is shown in Figure 2.

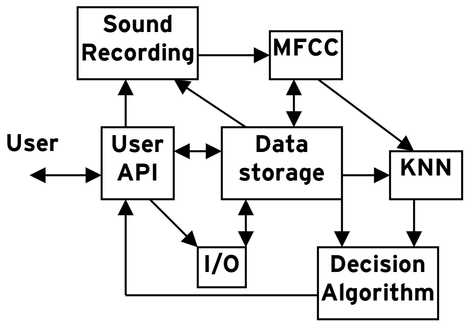


Figure 2: A functional block scheme of single user identification.

The API is divided (by functionality and access not by security) between designer/system administrator API and common user interface. The designer/ system administrator API is done via 2 configuration files (*config_knn.txt* –controlling recognition process and *config_param.txt* that is involved in the speech extraction, for examples of the configuration files please see Annex A) where almost all settings controlling the behaviour of an application can be set. It should be noted that in order the application function properly the parameters in both configuration files must be set by professionals. The user API is done via a console application menu whose outlook is as follows:

Speaker Identification Menu:

- Add and record new user press 1
- Identify unknown speaker press 2
- Display users in database press 3
- Remove a user from database press 4
- Exit press 5

Choice:

The operation flows are as follows. After choosing a desired option from the menu a particular code is invoked. When adding a new user first a name is require to be entered that must be unique. After that a speaker is prompted to speak for pre-set time given by a configurable parameter. There is also a minimal training time which must be met prior to saving new features. When an unknown individual is to be recognized, the recognition data are first updated and stored in operational memory. Then he or she is prompted to speak for a given time period (also set in the configuration file). If minimal length is not met warning is displayed. Then speech is transformed into MFCC and is passed to KNN algorithm which based on the stored training vectors chooses nearest neighbours and determines particular winners with their scores. Finally, all winners are considered in the concluding decision process whose output is provided to the user via a console window. The remaining options (listing and deleting users) are self-explanatory.

2.1.7. Algorithms and Input and Output description, Installation Guide, and User Guide

As the multi-user identification task is more general than the single user one, the designed algorithm and implemented system for multi-user scenario can be directly used (perhaps some parameters may differ if optimal performance is to be achieved), please see for the detailed descriptions deliverable D5.3.1 [21].

2.1.8. Conclusion

In the current application standard, successful and very popular methods are implemented so far. Their mutual optimization has been done in the context of the proposed kind of application using extensive search on a real speaker database, e.g. weighting windows, local distances, a speech detection algorithm (proposed), number of neighbours, frequency ranges, feature dimensions, etc. In this way accuracy and robustness improvements were observed compared to the classical settings. However, several extensions to know methods have been proposed and are being tested and evaluated for the final application.

In the next step additional and more advanced signal processing techniques may be tested and implemented, like: feature warping and mapping, model mapping (to suppress session variability), etc. It is also possible to implement higher level features like pitch period to increase the accuracy and robustness. In this case more classifiers may be employed and merged together by a fusion technique or a proper super vector concept [4] will be tested and if it is beneficial it will be implemented as well.

2.2. Face Recognition

Face recognition systems for personal identification are increasingly used in a wide range of applications. Development of recognition algorithms and methods made possible usage of identification and verification systems in commercial field. However these systems do not achieve comparable recognition rates under uncontrolled and unconstrained conditions. Face recognition under these conditions is still a challenging problem in spite of recent advances in well-constrained face recognition. Several recent releases of the Face Recognition Vendor Test (FRVT) [37] also indicate that the development in the area of face recognition under uncontrolled and unconstrained conditions is a still essential. The main aim of our research is to propose face recognition system, which has to be fully automated and users have to be able to control it. Our system also has to work properly under uncontrolled and unconstrained condition (e.g. varying illumination, face expressions or occlusions, varying face pose etc.). In comparison to several commercial face recognition systems, the added value of our system is an automatic training process, which uses clustering algorithm for selection of appropriate training samples. Based on this we can achieve higher recognition rates even under varying face pose during the identification of users.

2.2.1. Theoretical Basis

In this chapter we present our basic research in the face recognition area, which was described for more detail in [22] and [23]. We are focused on research in the field of face recognition under uncontrolled and unconstrained conditions (such as noise, partial occlusions and varying illumination) and we also evaluate suitable algorithms and methods for HBB-NEXT face recognition system. Selected methods are tested on the same databases (FERET [24], CMU PIE [25]) and under the same unconstrained conditions, which allow a clear comparison of the influence of various types of noise, partial occlusions and varying illuminations on the face recognition accuracy. In addition we explore these methods for cases where only a few (up to 4) samples are available.

The main task is to design a system that properly recognizes faces in real time and real conditions. We examined the real conditions in several ways: varying illumination, different face expressions, partially occluded images and noisy conditions. In our research we worked with selected face recognition methods:

- Principal Component Analysis – PCA [26]
- Kernel PCA – KPCA [27]
- Generalized Discriminant Analysis – GDA [28]
- Radial Basis Function Network – RBF [29]
- Support Vector Machine – SVM [30]
- Local binary patterns – LBP [31]

In [22] we provide a comparative study of several conventional face recognition methods (PCA, KPCA, GDA, SVM and RBF) that are suitable to work properly in our face recognition system. We evaluate the influence of varying illuminations and pose of faces on face recognition accuracy. This task is very important, because significant changes in illumination can usually lead to fail of face recognition system. Experiments with different illumination conditions were performed on the face database CMU PIE, which includes the images taken under just the flash, under just the room lights and with both of them. We concentrate on the ability of used methods to eliminate the impact of varying illumination. The best recognition rate under varying illumination was achieved by RBF method followed by SVM and GDA method. These methods can significantly eliminate the presence of varying illuminations in human face images, even in the case of basic (cropping and histogram equalization) preprocessing of input images. In case of pose changing, these methods were stable. On the other hand, PCA method and its extension KPCA achieved significantly lower recognition rate. We also compare training and testing time of used methods. The performance results indicate that, the training and testing of RBF method took much of the time. LBP method reached results that are suitable to deployment in real time. Based on these experiments, we can conclude that, the SVM and GDA methods best meet the requirements for our HBB-Next face recognition system.

In [23] we evaluate the influence of noise and partial occlusion on face recognition accuracy. We are focused on occlusions of eyes and eyebrows as these are the most significant features of a face. We also compare face recognition rates achieved by machine learning methods with accuracy achieved by human perception. Noise in human face images can seriously affect the performance of face recognition systems. It can be produced by the sensor of a scanner, digital camera, or a by-product of image transmission, image copying etc. Noise decreases the information value of useful data. Another issue with image-based face recognition in non-standard conditions is aimed at partially occluded face images. Some parts of face, such as eyes, nose and mouth have been typically indicated as very important in research of human vision and recognition by humans. In our experiments we evaluate impact of various partial occlusions of human faces on several face recognition methods. One facial feature has received little attention from researchers in face recognition area - the eyebrows. We included reference PCA method, kernel based methods, such as KPCA, GDA, SVM and RBF neural network. These selected methods were evaluated on the FERET face database. Original FERET images were modified by various partial occlusions and noises. Our experiments show that SVM and GDA methods are ones of the most efficient and stable methods in unconstrained conditions. RBF method achieved the worst recognition rate when partial occlusions were included. On the other hand, the best results came from RBF method in case of tests with original (non-corrupted) face images.

In several our experiments (such as [32]) the LBP method for feature extraction was implemented. We achieved the best results, when we used LBP method to feature extraction and SVM method as a classifier. This approach best meet the requirements for the HBB-NEXT face recognition system (real time recognition in real world condition).

Future work:

Our HBB-NEXT face recognition system works in two main phases. The first phase is a training process of our system and the second is classification of users. We have used a Kinect as an input sensor for both of them. Kinect is able to take up to 30 photos of scene per second. In case that our faces detector works flawlessly and Kinect can takes 30 images per second, our recognition system can collects up to 300 face images per 10 seconds.

Based on our experiments we can conclude that tested methods achieve reliably results when up to 4 images per subject are available for training process. In the last release of our HBB-NEXT face recognition system we have to manually select an appropriate face images for each subject from a folder where our system stored all face images taken from scene. So our research in the field of automated selection of training samples is essential for HBB-NEXT face recognition system.

Clustering algorithms for selection of training samples and automated training process are the topics of our research in nowadays.

2.2.2. Algorithms description

In this section we describe architecture of the proposed system and workflow used in the HBB-NEXT project. Since the face recognition task requires processing of the input from a camera, we decided to use a pipeline model where each frame captured by the camera is processed by set of modules. These modules contains an implementation of the chosen face recognition method and, together with the camera, operates in a loop.

The main recognition process (the system loop) consists of following sub processes:

- ***Image acquisition*** - reads an image from the camera, converts it to the system format and pass it to the system pipeline.
- ***Face localisation*** - localizes the faces in the image on the image and associate found coordinates with the image. For the purpose of this project we use two different localisation implementation:, depending on the camera which is used:
    - *OpenCV Cascade detector* - this is used with ordinary cameras with standardized interface (e. g. web cameras compatible with OpenCV library)

- *Kinect Toolkit Face detector* - this detector is used with the Kinect Camera[1]

  ♦ ***Pre-processing of localised faces*** - copies regions of the image containing faces into new samples, converts the regions into the appropriate resolution and the appropriate colour format.

- ***Feature extraction*** - Extracts features from preprocessed faces. For the purpose of this project we decided to use LBP histograms as features. LBP histograms are considered as one of the best features for recognizing faces ([33], [34]) even when only a limited number of samples is available [32] and can be easily computed in the real time [35].

- ***Classification of faces*** - For the classification of features extracted from faces we propose to use two methods depending on the number training images and number of identities which is to be used within the system:

  - Support Vector Machines - is used when only relatively small number of identities is considered in the system. Main disadvantage of this method is the time-consuming training of the model when large number of samples is used.

  - Nearest neighbour distance matching (with the use of Chi-square distance) - this algorithm can be easily parallelised and used in distributed system. The training is done simply by inserting features into the database.

- ***Face tracking*** - In our implementation we localize only frontal faces in the image because the vast majority of face recognition methods is reliable only with use of frontal face images. Once the face has been recognized, it is tracked, what significantly saves computational resources and can follow the subject even after changes in pose. In our system we use TLD tracker proposed by Zdenek Kalal [36].

[1] Face tracking SDK is a part of Kinect for Windows Developer Toolkit v1.5 - http://www.microsoft.com/en-us/download/details.aspx?id=29865

- ***Temporal filtering*** - localisation of faces is not always reliable due to changes in environment, poses or imperfection of cameras. To prevent the system from triggering events in case of every detection or tracking failure we propose a method for filtering the most frequent types of failure. For instance to filter out short focus-losing windows in tracking.
- ***Event trigger*** - this module sends a notification of user state changes to the rest of HBB-NEXT applications.

The core of the face recognition system is written in C++ with using OpenCV library. We decided to use an open source project Biosandbox developed at STUBA. In this project a large number of required algorithms was already implemented in form of dynamic modules.

2.2.3. Input and Output Description

The basic concept of the Biosandbox system is shown in Figure 3. The composition of modules is defined in an XML configuration file that can be created in a graphical editor that is also part of this project. For the purpose of the project we developed 3 additional modules:

- Kinect Input module - reads images captured by Kinect camera.
- Tracking module based on TLD [36] - once a face is localized in the image, this modules track its position even when the pose of the subject has changed.
- Temporal filtering module combined with event trigger - checks presence of the subject in the scene with considering detection results and previous results in the time.

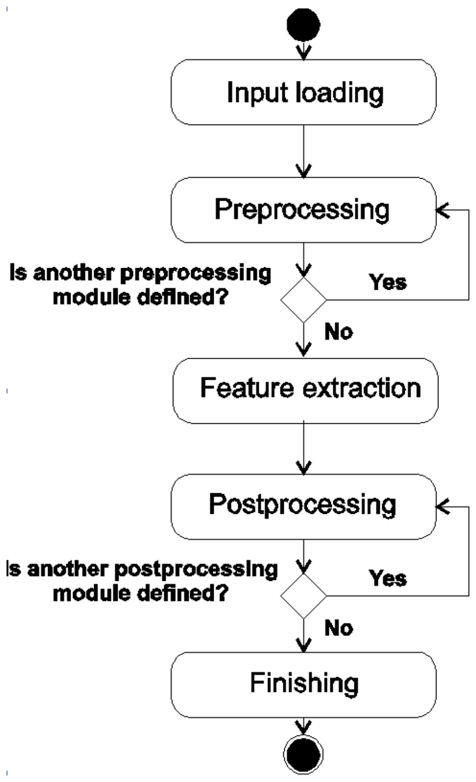

Input loading
Preprocessing
Is another preprocessing module defined?
Yes
No
Feature extraction
Postprocessing
Is another postprocessing module defined?
Yes
No
Finishing

Figure 3: Schematic illustration of processes sequence during system execution.

The face recognition module sends XML output to a server. The XML describes the identity of subject (person), probability of correct classification, presence of a person in the room (present, idle, quit) and it contains all the identities that were trained. The example of using the XML structure is shown in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<application>FACE_RECOGNITION</application>
<users>
      <user>
            <name>John</name>
            <probability>10</probability>
            <presence>PRESENT</presence>
      </user>
      <user>
            <name>George</name>
            <probability>75</probability>
            <presence>IDLE</presence>
      </user>
      <user>
            <name>Bill</name>
            <probability>60</probability>
            <presence>QUIT</presence>
      </user>
</users>
```

Figure 4: The example of XML structure

2.2.4. Installation Guide

System requirements:

Supported operating systems: Windows 7, Windows 8, Windows Embedded Standard 7

Software requirements:

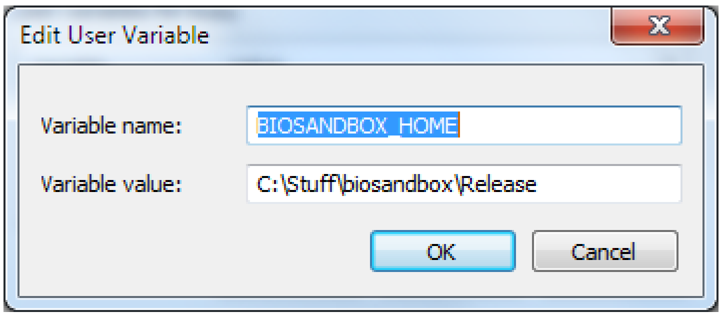
The Kinect for Windows Software Development Kit (SDK) v1.6 or later required

Hardware requirements:

- 32-bit (x86) or 64-bit (x64) processor
- Dual-core 2.66-GHz or faster processor
- Dedicated USB 2.0 bus
- 2 GB RAM
- A Microsoft Kinect for Windows sensor

Installation guide:

1. Extract Biosandbox.zip file to HDD
2. In System properties create new environment variable BIOSANDBOX_HOME, which contains path to extracted application:



Edit User Variable

Variable name: BIOSANDBOX_HOME

Variable value: C:\Stuff\biosandbox\Release

Figure 5: Creating new environment variable BIOSANDBOX_HOME

2.2.5. User Guide

Our HBB-NEXT face recognition application works in two phases. The first phase is a training process and the second is recognition of users.

Training process:

- Create subfolder "faces" to application folder
- Run "capture.bat" to capture training samples. User has to stay in front of Kinect. Camera captures images, when the user has red square over the face. These images are saved to subfolder "faces".
- From the captured faces you have to select 5-10 training images per subject (from subfolder "faces")
- Move selected images to application folder and rename them to followed format "*XXXXXYY.png*", where XXXX is No. of identity and YY is No. of sample.
- Save the list of selected trained images to "*train.txt*", *example:*

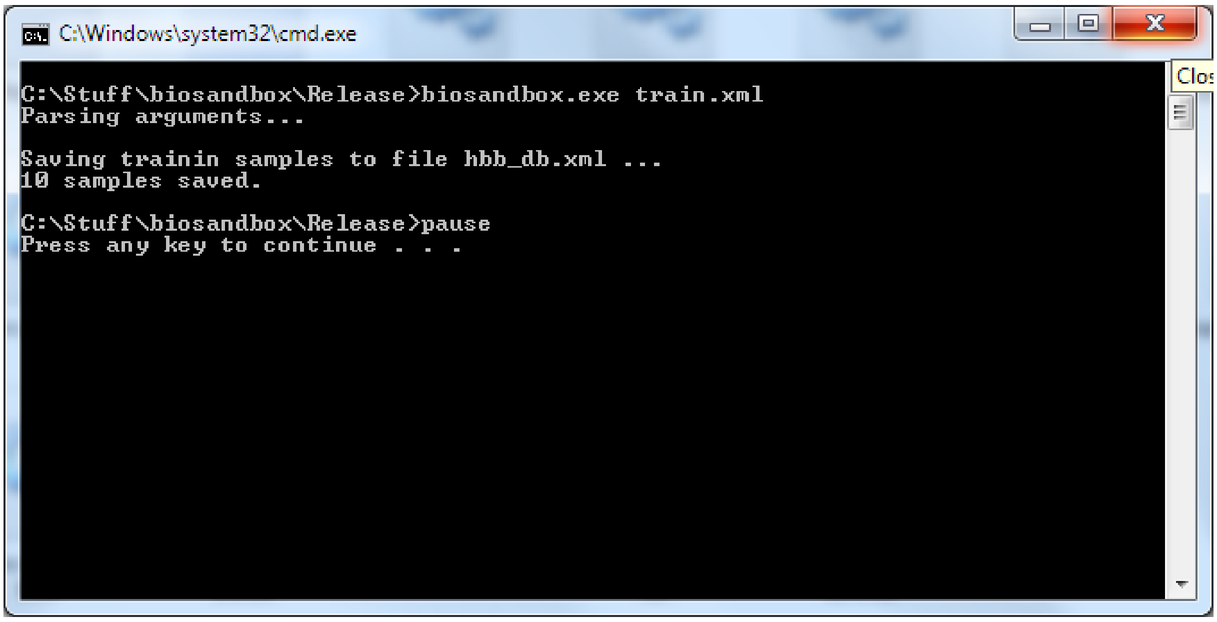
```
0000101.png
0000102.png
0000103.png
0000104.png
0000105.png
0000201.png
0000202.png
0000203.png
0000204.png
0000205.png
```

- Create file "*names.txt*", with real name of user in followed format:

  XX ZZZ, where XX is No. of identity and ZZ is real name of user, which will send to server.

```
-1 <unknown1>
0 <unknown2>
1 John
2 Michael
```

- Run "*train.bat*" to train face recognition application.



```
C:\Stuff\biosandbox\Release>biosandbox.exe train.xml
Parsing arguments...

Saving trainin samples to file hbb_db.xml ...
10 samples saved.

C:\Stuff\biosandbox\Release>pause
Press any key to continue . . .
```

Figure 6: Training output in command line

- In configuration XML: "*recognize.xml*" set IP address and port of HBB server:

  <Module src="TldTracking" namesFile="names.txt" server="127.0.0.1" port="27015"/>

Recognition

- Run "recognize.bat"

2.3. 3D Face Recognition

HBB-NEXT project is an idea where is very important to keep trust because of users will use many applications which will be used for various purposes. For example: logging to trust application, verification for paying or buying products, etc. We have included to HBB 3D face recognition for High level verification that could increase the security of some used applications. The main target of this part is to create a module that will be able to recognize human faces from depth data.

Face recognition is intensively studied problem in computer vision community. It is a natural and nonintrusive way of identification and authentication. This domain has been researched for many years and the effort of everybody who ever worked on this problem is to make computers identify people in the same way as people identify each other. We distinguish each other by our physical characteristics especially by our face.

In general there are three main approaches based on type of data which are used in the recognition process. We know methods based on 2D intensity image, 3D facial data and the technique which is using both types of data.

2.3.1. Algorithms

The whole process of recognition consists of 3 main stages. The first one is acquisition and pre-processing, the second is data registration and the third stage is recognition. We currently accomplished the first and part of the second stage in which we implemented automatic landmark localization. The current state of our research is discussed closely in the next chapters.

2.3.1.1. Data Preprocessing

The goal of pre-processing stage is to get cropped depth image of face in the best possible quality. The first step is to obtain the position of the user head in the image. We decided to use Face tracking toolkit that is included Kinect development package.

2.3.1.2. Face Tracking

Microsoft Kinect SDK offers Face Tracking Toolkit. This toolkit can track human faces under prescribed conditions. Face Tracking Toolkit offers 120 tracked points on face. Each point has its own meaning. This is very useful functionality which we use in more stages of the recognition process.

2.3.1.3. Conditions for Face Tracking Toolkit

If face has to be tracked, some conditions have to be accomplished. The external infrared light illumination has bad influence to tracking performance. Kinect is emitting the IR light to obtain the depth information from scene that’s why strong external IR sources like a direct sun light limit the Kinect functionality. The other condition is about distance from the camera. While the distance of the human face is between 80 cm and 90 cm face is in the result image bigger and points more accurate. Figure 7 is example of tracked points on face.

Face Tracking Toolkit from Microsoft Kinect SDK requires some inputs. The three main inputs are:

- Colour Stream: This stream comes from RGB camera. It is same stream as in usual cameras.

- Depth Stream: Depth Steam contains the distance (in meters) from the camera plane to the nearest object. Distance is measured by infrared ray.
- Skeleton Stream: Skeleton stream contains information about persons in front of Kinect. Each person has own index. This stream is helpful for finding head position of human being.

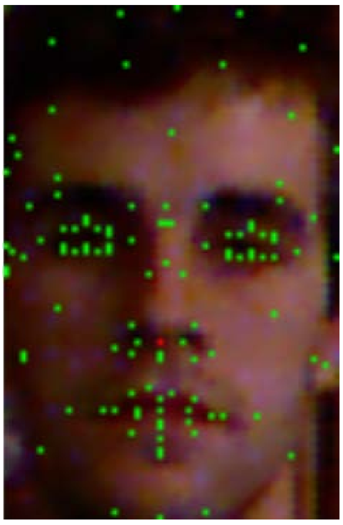


Figure 7: Illustration of 120 tracked points of face with Face Tracking Toolkit

2.3.1.4. Meaning of tracked points

Each point has own meaning. Face contains 87 basic and 20 complementary points. Each tracked point has own ID number, which belongs to some part of detected face. If some point could not be detected then position of this point is approximately calculated by Kinect SDK. Each tracked point on detected face has three coordinates - X, Y, Z. The origin is located at the camera's optical centre, Z axis is pointing towards a person, Y axis is pointing up (based on a right-handed coordinate system). The measurement units are meters. There are three very dense fields of points around left and right eyes and mouth. These coupled points have ascending numbering of own index.

2.3.1.5. Face Extraction

The result of extraction is depth image of the face. Face points (mentioned above) positions are mapped into colour image. We need distance from depth image too. Because of that conversion to depth image is necessary.

2.3.1.6. Calculating Angles

Because we want to cut out the face from the image due to face recognition, it is needed to have approximately the same location of head of scanned person. This goal we provide primary by calculating angles of scanned head. As the head can be rotated in three different directions:

- Pitch: It is a vertically rotating head, from bottom to top, or vice versa. This rotation generally confirms some kind of information.
- Yaw: It is horizontal rotating head, from left to right, or vice versa. This rotation usually means expressing disapproval some kind of information.
- Roll: It is rotating head about Z-axis, which is based directly from my eyes.

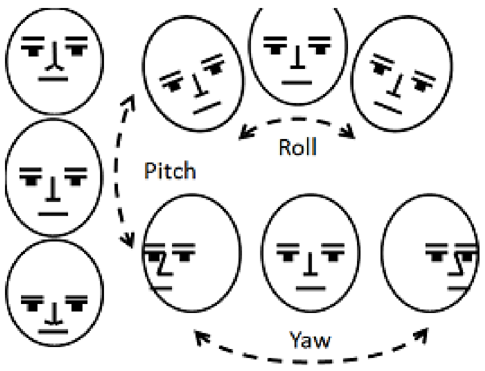


Figure 8 Three possible rotations of head

Kinect SDK calculates correct angles if the person is directly in front of the sensor. We have the function which calculate angle from stored points. This function calculates good angles if person is not directly in front of sensor. For calculation of angles are used always only two points from 120 points. Firstly, difference between each axis of these two points is calculated. Then each difference is raised to power of two and then the sum of these three powered numbers is created. Finally this sum is square of root two and by division and invert tangent function we get rotation angle. Pitch and roll is calculated from points 0 and 9. For yaw are used points 120 and 116. When all of three angles are in specified range than the face is captured.

2.3.1.7. Cropping the Face from Image

Only face is needed from whole captured image. Because of that cropping of face is needed. Each face is in the rectangle, which consists of 4 points on the head. The side edges consist from points which position is most left and right. The highest point makes upper edge and the lower edge consist of the lowest point. Then the cropping is based on this rectangle made by these 4 points.

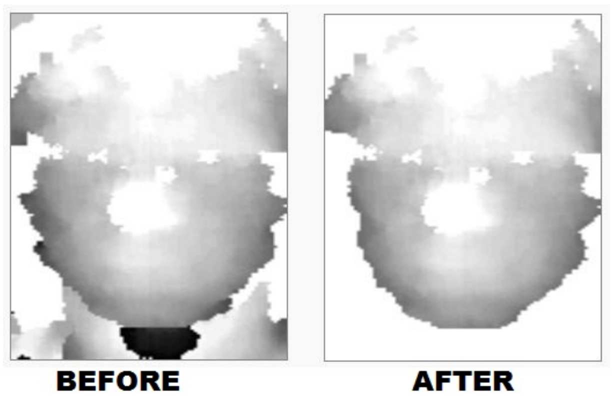


Figure 9: Illustration of cropped face from one frame of depth stream

2.3.1.8. Averaging and Smoothing Face Image

The depth output from Kinect camera is noisy and it contains holes in the image (see Figure 10). To suppress these negative factors we use some techniques to make the output better.

Firstly we implemented a hole-filling algorithm, which computes the missing data according to known adjacent depth data. This is applied for filling the small gaps in the image.

We also used the averaging which takes several depth frames in the one row and computes the average from known depth data only. The principle is very simple and it can be upgraded to be more robust. The problem is when user during the phase of collecting the number of frames slightly moves his head. For example if we have frame rate 30 fps we can take 10 face images and it takes about 0.33 second which is quick but it is still enough time for user to change his position with the result of ruining the averaging process. An enhancement would represent the reset of collecting phase when movement of the head is detected or aligning collected frames to the same position.

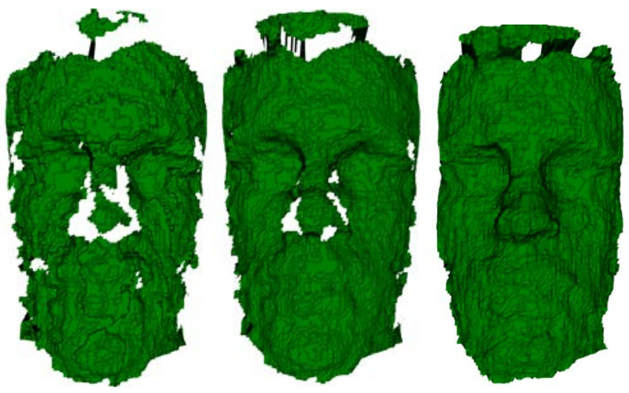


Figure 10: Examples of scanned faces from left: 1. Face created from one depth image, 2. Face created by averaging multiple images, 3. Averaging multiple images and gap filling algorithm

2.3.1.9. Automatic Landmark Localization

Our system in a current stage has implemented basic landmark localization. As we mentioned above Kinect SDK offers the face tracking toolkit which provides good starting position. Toolkit provides functionality for tracking 120 points on face which are mapped into colour image. We converted all tracked points to depth image. We found that tracked points match the positions in the depth face image with a noticeable difference. So correction of points positions are necessary. For the purpose of aligning scanned faces in recognition process we don't need to tune position of all 120 tracked points, but it is sufficient to choose few key points in our case 4 (nose tip, nasal bridge, eyes) and find the correct position in depth image.

2.3.2. Inputs and outputs

2.3.2.1. Inputs

The input of this application is depth stream from Kinect sensor that is used for obtaining face data. Application uses the highest resolution of depth camera (640x480) because of accuracy.

2.3.2.2. Outputs

Actually the main output is visualisation of pre-processed 3D face data, showing of landmarks. The application does not have a communication module yet.

2.3.3. Installation

1. Local host with local host mysql (WAMP server recommended).

2. Create new database with tables, all is in the localhost.sql that is needed just execute.
3. In the Default Root of Visual Studio "//projects" extract .rar with kinect files. Use the following structure <Visual studio root>/projects/kinectfiles/<username>
4. Turn on the localhost with database.
5. Open Visual Studio, needed to have opened instance of the Visual Studio, it is not necessary any Visual Studio project, just opened Visual Studio.
6. Open "kinectapp.exe".
7. In the application window choose option "localhost".
8. Browsing application functionality.

2.3.3.1. System Requirements

Hardware:

- 32 bit (x86) or 64 bit (x64) processor
- Dual-core 2.66-GHz or faster processor
- Dedicated USB 2.0 bus
- 2 GB RAM
- A Kinect for Windows sensor, which includes special USB/power cabling

Software:

- Microsoft Visual Studio 2010 Express or other Visual Studio 2010 edition
- .NET Framework 4.0
- Localhost with mysql
- Created folders for all users <Visual studio root>/projects/kinectfiles/<username>

2.3.4. User guide

Currently, the application is able to create user profiles in the database and stores facial data of users; it can localize facial landmarks and contains first effort of recognition phase. The application is not able to recognize yet but first pre-processing shows good vision of recognition.

2.3.5. Conclusion

The goal of our research is to implement recognition functionality. Actually, we focus on finding and modifying recognition algorithm suitable for 3D facial data we obtained. After successful implementation of recognition we plan to create an application for testing purposes. Successful testing will lead to integration the system into HBB-NEXT project. The purpose of our system will be to recognize people in a household where HBB-NEXT system is meant to be used.

We plan to focus on recognition phase and research recognition algorithm that can be used for our purpose, because we have met couple problems such as low accuracy, noise, etc. we would like to provide in the next phase a recognition algorithm fit for our purposes. We would like to reduce researched problems and continue with developing of recognition algorithm.

2.4. Iris Recognition

Iris is one of the most popular biometric traits. Combination of touchless scanning, stability over time and high recognition accuracy enable the use in surveillance as well as security applications. J. Daugman [38][39] proposed a method which reaches very high recognition accuracies (almost 100%, depending on the used dataset).

It was shown that the recognition accuracy depends on the quality of the captured iris image and image preprocessing. To reduce the negative influence of illumination NIR (near infrarred) light sensing camera is used. Using NIR light allows us to add addition light source without influencing comfort of sensing.

In this chapter we describe details of the iris recognition system implementation for the HBB-NEXT project. Within this system is the iris recognition task used for securing access to personal information and confidential connections (e.g. remote banking or a government related interaction). The iris recognition system works in the verification mode and processes a single image from camera. This image is taken upon request from a user when secure access is needed.

2.4.1. System Setup

In our experiments we use AVT Guppy F-038B NIR monocular camera. This camera has the Sony ICX418 CCD interlaced sensor, which operates at 30fps at full image resolution of 768 x 492 pixels.

These attributes fully meet requirements of the HBB-NEXT recognition systems. Moreover we chose a NIR type of the camera with a sensor sensitive in near infra-red range (NIR) of the electromagnetic spectrum. The main advantage of systems using NIR spectrum lies in ability to penetrate dark-coloured irises, the dominant phenotype of the human population, revealing texture not easily observed in the visible wavelength spectrum.

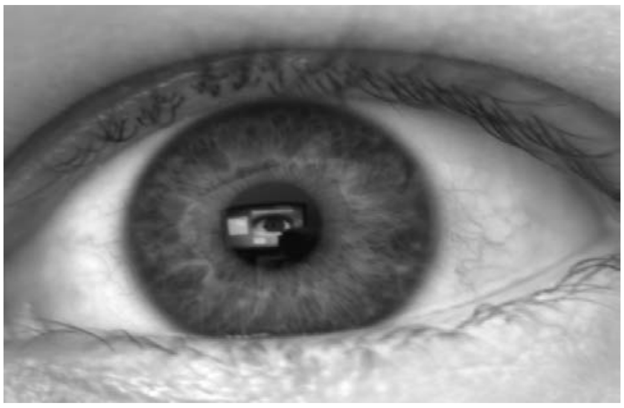


Figure 11 An example of a picture taken with the Guppy F-038B NIR camera.

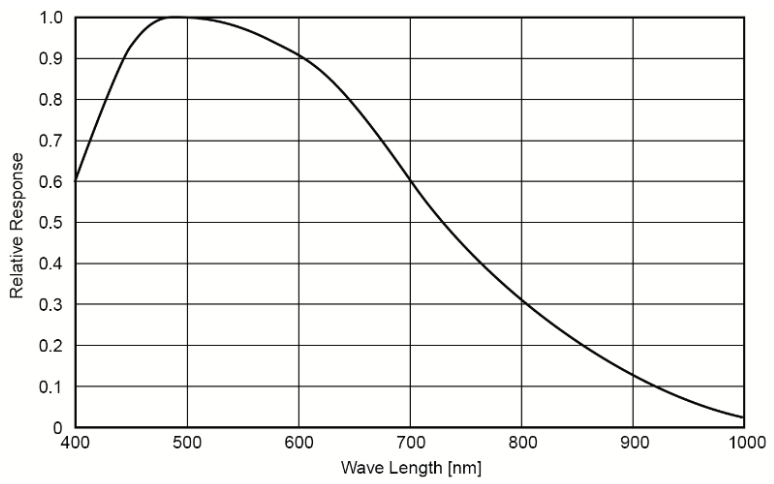


Figure 12 Spectral sensitivity - Guppy F-038B NIR camera without cut filters and optics.

The camera operates at four main modes as described in following table by assuming that bus speed is 400Mbit/s. This camera uses Firewire IEEE 1394a interface.

Mode	Resolution	Color mode	Max. frame rates
0	**768 x 492**	**MONO8**	**@30 fps, 2 x V-binning, interlaced, field integration mode**
1	**768 x 492**	**MONO8**	**@30 fps, no binning, interlaced, frame integration mode**
2	**384 x 244**	**MONO8**	**@59 fps, 2 x full binning for aspect ratio, non-inter-laced, progressive readout mode**
3	**768 x 244**	**MONO8**	**@59 fps, 2 x V-binning, non-interlaced, progressive readout mode**

Table 3 Operating modes of the Guppy F-038B NIR camera.

2.4.2. Algorithm

The general iris recognition process consists of two parts. First, an iris and eyelids is localized and segmented. Second, an iris code is extracted from the iris texture. Afterwards the code is verified according to claimed identity from the database.

2.4.2.1. Image Acquisition and Iris localisation:

We use the C interface of Vimba AVT SDK to capture images from a camera. These images are then deinterlaced and provided to the segmentation subsystem. Segmentation algorithm uses Hough transform [40] to localize inner and outer boundaries of an iris and also to localize both eyelids afterwards. The localisation algorithm consists of following steps:

1. Equalization of histogram in the iris image. In this step the intensity band of the captured grayscale image is transformed to the intensity band required by following steps in the localization and the feature extraction. Since image is taken under the NIR spectrum it consists of limited band of intensities which may not be suitable for following processing.
2. To the equalized image of the iris is applied threshold and image is binarized. This thresholding helps to localise inner iris boundaries. The Hough transform is used to localise circle in the binarized image.
3. We apply the Hough transform to the image from step 1 in order to find outer boundaries of the iris. The algorithm searches the image for significant circular objects. The outer boundary is chosen as a trade-off between confidence value from Hough transform and distance from the centre of the pupil (which was localised in the previous step).

2.4.2.2. Iris coding and template comparison:

We use the general iris recognition algorithm proposed by John Daugman [39] which is using Gabor filters. These filters are implemented as a set of convolution kernels which is used to extract iris code. After convolving the extracted iris texture, the response is transformed into iris code by applying following rules:

- If point in response is lower than zero then 0 is written to the iris code.
- If point in response is greater than zero then 1 is written to the iris code.

The result is a bit code which is used as a template and stored in the database. Recognition is done by searching the database for the most similar code. Similarity is defined as a minimal Hamming distance between iris codes.

2.4.3. Conclusion

We are currently integrating the iris recognition subsystem to the HBB-NEXT system. In the future we plan to investigate the possibility to optimize convolution kernels to improve recognition rates in comparison to the original method proposed by John Daugman.

2.5. Section Conclusion

As it was mentioned in the previous sections the presented solutions only exist in the form of early demos. All of them, however, are already capable of interaction with the rest of the multimodal system designed by STUBA, even though there is still room for improvements, particular to each modality. Further work in the field of multimodal user identification and authorisation will include, apart from improving the algorithms themselves, gradual integration of the modalities' outputs to enhance the quality of proper identification.

We have developed a mechanism for integration of all multimodal interface modules into one integrated module that can be used as a final product for user-computer communication.

Additionally, a goal exists to create multi-stage user identification, consistent with the project's Work Package 5.

3. Trust and Reputation Module

Regarding the trust and reputation enabler, D3.2 presented a high level design indicating the data structures to be used, the sub-modules composing the trust and reputation module, the interfaces used to access the functionality exposed by such enabler as well as some sequence diagrams indicating the messages needed to be exchanged in order to fulfil the main processes, or use cases, that make use of this enabler, namely: U.027, U.031 and U.032 (see D2.1 [20]).

In this document, instead, after actually implementing the first version of the trust and reputation enabler, we present an updated version of the data structures finally used, as well as the interfaces actually implemented. Moreover, the new trust and reputation management algorithms developed within HBB-NEXT are also described, preceded by a section stating which the actual progress with regards to the current state of the art in this field is. Finally, some indications on how to actually use the developed software are also included.

3.1. Theoretical basis (progress beyond SoTa)

Reputation management systems for distributed and heterogeneous environments have been studied since a while [41], [42]. Moreover, reputation frameworks have been proposed in different contexts, e.g., P2P file sharing [44] and reputation enabled service-oriented frameworks [48], [49]. Thereafter, the next trends that we could analyse are the application of reputation frameworks for enhancing hybrid broadband broadcast systems and the proposal of new reputation frameworks advocated to these environments [51], [52].

In D3.1 [18] we already analysed two reputation management solutions for Google TV [53], [54], [57] and Apple TV [55], [56], respectively. To these two prominent solutions, we should add some others recently rising with strength, such as Samsung Smart TV [58]. Yet, whilst those solutions, amongst others, have achieved an unquestionable success in terms of users' acceptance [50], we believe that the underlying trust and reputation management mechanisms deployed within their corresponding app ecosystems have not received the proper attention [45], [46], [47].

Neglecting the importance of an accurate and robust trust and reputation management mechanism (as well as resilient to certain specific threats [59]) within an appstore could lead to irreparable leakage of private and sensitive users' information or even a real harm in the devices where the applications offered by such appstore are installed and/or executed [60].

Despite the reluctance of these major app platforms to provide technicality about their reputation systems, authors of [61] extracted the five most common reputation models used by current main providers of online products and services (not only app providers), based on the observance of the way feedback is collected from costumers and how reputation scores are calculated and distributed [18]. These five models are: i) "vote to promote", ii) "Content rating and ranking", iii) "Content reviewing and comments", iv) "Incentive points" and v) "Quality karma".

Factually, any of those reputation management solutions offer customized or user-tailored reputation scores. Thus, as we will see later, we have developed a trust and reputation management mechanism able to provide such user-tailored scores which, to the best of our knowledge, is novel within the context of hybrid broadband broadcast apps ecosystems.

3.2. Algorithms description

As stated in D3.1 [18], "The key factor in the design of this module [the Reputation Computation Engine] consists of implementing the right reputation function for aggregating feedback values and calculating a final reputation score". Hence, we have developed and studied several reputation computation engines as part of the trust and reputation enabler.

The results of each computation engine depend on the aspects or elements taken into account when performing the calculation, like users' preferences, amongst many others. Furthermore, the computation engines not only differ on the way of performing the reputation calculation but also in the resources they need to work. The aim of this section is introducing different computation engines in order to analyse the feasibility of each of them under certain system conditions.

3.2.1. Average

The first and most straightforward computation engine that we describe is named *Average*. Being the feedback provided by the Issuers (end users in the context of HBB-NEXT) a real number belonging to the same interval ([0,1], for instance), this engine computes the reputation value for each Subject (HBB application in the context of HBB-NEXT) as an arithmetic mean according to the formula shown in Equation 1.

$$Rep(App_i) = \frac{1}{n} \cdot \sum_{j=1}^{n} feedback_{u_j}(App_i) \quad (1)$$

where

$Rep(App_i)$ → Represents the reputation of the i-th HBB application

n → Represents the total number of end users in the system

u_j → Represents the j-th end user

$feedback_{u_j}(App_i)$ → Represents the feedback provided by the j-th end user regarding the i-th HBB application

3.2.2. Weighted Average

This computation engine extends the previous one by assigning a weight to each Issuer, so the reputation values are computed performing a weighted average accordingly. The weights are defined according to the estimated goodness of each issuer, in order to associate a lower importance to the recommendations given by malicious users when computing reputation values.

The goodness (i.e. the associated weight) of each user is calculated from the deviation of his/her recommendations with regards to the recommendations of the rest of users. For instance, a user will decrease her goodness if she provides low value recommendations while the rest of users provide high value recommendations to the same Subject (HBB application), or vice versa. To do so, an initial weight is given to each user, and those weights are updated whenever a user provides a new recommendation, by means of the formula shown in Equation 2.

$$\mathrm{w}_{\mathrm{u_j}} = \mathrm{w}_{\mathrm{u_j}}^{1/_{1-\mathrm{d}}} \cdot \left(2 - \mathrm{w}_{\mathrm{u_j}}^{1/_{1-\mathrm{d}}}\right) \qquad (2)$$

where

w_{u_j} → Represents the weight given to the j-th end-user

$d = \left| Rep(App_i) - feedback_{u_j}(App_i) \right|$ → Represents the deviation from the overall reputation score of the i-th HBB application with regards to the feedback given by the j-th end-user

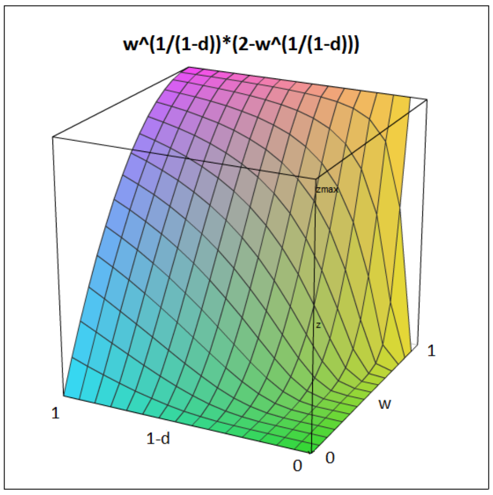


Figure 13. Plotting of Equation 2, depicting the weights updating in the 'Weighted Average' reputation computation engine

As we can observe in Figure 13, the weights w_{u_j} are properly calculated since

$$(1-\mathrm{d}) \to 0 \Rightarrow \mathrm{w} \to 0 \text{ and } (1-\mathrm{d}) \to 1 \Rightarrow \mathrm{w} \to 1 \tag{3}$$

In other words, the smaller the deviation (i.e., $d \to 0$, ergo $(1-d) \to 1$), the higher should be the updated weight for that particular end user, and vice versa.

Finally, the reputation value of the i-th HBB application is computed as a weighted average given by the formula presented in Equation 4.

$$\mathrm{Rep}(\mathrm{App_i}) = \frac{\sum_{j=1}^{n} \mathrm{w_{u_j}} \cdot \mathrm{feedback_{u_j}}(\mathrm{App_i})}{\sum_{j=1}^{n} \mathrm{w_{u_j}}} \tag{4}$$

3.2.3. Limited Weighted Average

As the reader can easily observe, the previous two computation engines calculate the reputation score for an HBB application taking into account all the feedbacks provided by previous users. While such option makes the computed reputation scores more accurate, it is obviously not scalable when the number of previous feedbacks is significantly high. This issue becomes even more critical when the device computing the reputation score has limited or constrained capabilities (in terms of computation, storage, etc).

In order to tackle this problem we propose the *Limited Weighted Average* reputation computation engine, which only takes into account the m most recent feedbacks when computing the reputation score of an HBB application. Hence, such parameter m should be chosen to be small enough to avoid overloading the reputation computation engine, while at the same time it should be big enough to provide accurate results.

Formally, the reputation in this computation engine would be calculated according to the formula shown in Equation 4**Chyba! Nenašiel sa žiaden zdroj odkazov.** (bear in mind that here w_{u_j} is updated according to the formula shown in Equation 5).

$$\mathrm{Rep(App_i)} = \frac{\sum_{j=1}^{\min(n,m)} w_{u_j} \cdot \mathrm{feedback}_{u_j}(\mathrm{App_i})}{\sum_{j=1}^{\min(n,m)} w_{u_j}} \quad (5)$$

3.2.4. Preferences Weighted Average

The computation engines shown so far calculate global reputation values representing the opinion of all users as a unique value. However, they do not take into account that each user could have a different opinion about the same service, evaluating it with different values, thus expecting different reputation values. In order to provide customized reputation information adapted to each user, we should be able to compute user-tailored values depending on the similarity between users.

In order to measure such similarity, this computation engine takes into account the user‘s preferences. These preferences express the assessment of each user with regards to the properties describing the Subject (HBB aplication). Hence, when computing the customized reputation scores for the user Alice, for instance, this computation engine gives a higher weight to those recommendations coming from those other users whose preferences match with the Alice‘s ones.

The assessment of the preferences by each user (how important/relevant each preference is for each user) could be done either when users are registered in the system, or the first time the user needs to obtain his/her customized reputation scores. For instance, in the case of HBB applications the user could establish her predilection regarding parameters like their price, usability, responsiveness, etc.

The *Preferences Weighted Average* computation engine establishes weights based on the similarity of the user preferences when calculating the reputation for a specific user. Being $pref_i$ the set of preferences of the user i, this engine defines the similarity of such user with regards to user j (denoted as $sim_{i,j}$) as the similarity/deviation of their respective preferences, as shown in Equation 6.

$$\mathrm{sim}_{i,j} = \sigma\left(\mathrm{pref}_i, \mathrm{pref}_j\right) \qquad (6)$$

Thus, the reputation for the user k, regarding the i-th HBB application, is given by the formula described in Equation 7.

$$\mathrm{Rep}_{u_k}(\mathrm{App}_i) = \frac{\sum_{j=1}^{n} w_{u_j} \cdot \mathrm{sim}_{i,j} \cdot \mathrm{feedback}_{u_j}(\mathrm{App}_i)}{\sum_{j=1}^{n} w_{u_j} \cdot \mathrm{sim}_{i,j}} \qquad (7)$$

3.2.5. Reputation computation engines analysis

In this section we analyse the reputation computation engines described above with regards to the following system conditions:

- **Number of users**: This parameter indicates the amount of end users participating in the system required for the reputation computation engine to provide accurate reputation scores.
- **Users' participation**: This parameter specifies how participative the users need to be within the system for the reputation computation engine to provide accurate reputation scores. In other words, it indicates whether the users participating in the system are being active and frequently querying the reputation scores of the target HBB applications as well as providing feedbacks for the installed ones.
- **Network resources**: It indicates how many network resources (in terms of bandwidth, for instance) are required for the reputation computation engine to have an optimal performance.

- **Computation resources**: It indicates how many computer resources (in terms of computation capacity, storage, etc.) are required for the reputation computation engine to have an optimal performance.

Computation engine	Number of users	Users' participation	Network resources	Computation resources
Average (Section 0)	low	low	low	low
Weighted Average (Section 3.2.2)	medium	medium	low	medium
Limited Weighted Average (Section 0)	low	medium	low	medium
Preferences Weighted Average (Section 3.2.4)	very high	very high	high	very high

Table 4. Reputation computation engines comparison regarding certain system conditions

Table 4 shows the minimum requirements of the four analysed reputation computation engines, with regards to the four system conditions described above, for the former to obtain an optimal performance.

Strictly following only such criterion, the "Average" reputation computation engine would be the most suitable one, since it is the one posing the minimum requirements. On the other extreme, we would find the "Preferences Weighted Average", as this reputation computation engine is the most restrictive in order to obtain an optimal performance.

Nevertheless, we would be sticking to just one side of the coin if we based our decision of which the most suitable reputation computation engine is, only on this criterion (the system conditions). In our opinion, it is necessary to take also into account a set of so-called performance measurements in order to better discern which reputation computation engine to actually use. The performance measurements that we used for our analysis are presented next:

- **Accuracy**: This measurement indicates how similar the computed reputation score with regards to the actual trustworthiness of the corresponding HBB application is.
- **Users' satisfaction**: This measurement indicates the similarity between the reputation score provided by the framework to a user (regarding a concrete HBB application) and the actual satisfaction or feedback of that user (with that specific HBB application).
- **Adaptability**: This measurement indicates the ability of the reputation engine to quickly and accurately react to sudden changes in the behaviour of a given HBB application (maybe due to some updates, for instance), by re-calculating an appropriate new reputation score.
- **Behaviour with malicious users**: This measurement indicates the level of resilience of the analysed engine with regards malicious users (providing ill-intentioned feedbacks, for instance).

Computation engine	Accuracy	Users' satisfaction	Adaptability	Behaviour with malicious users
Average (Section 0)	medium	poor	medium	poor
Weighted Average (Section 3.2.2)	good	medium	good	medium
Limited Weighted Average (Section 0)	medium	medium	good	medium
Preferences Weighted Average (Section 3.2.4)	good	good	good	good

Table 5. Reputation computation engines comparison regarding certain performance measurements

Table 5 shows the behaviour of the four analysed reputation computation engines with regards to each of the performance measurements described above. As we can observe, the "Average" reputation computation engine is actually the one outputting the poorest performance, whilst the "Preferences Weighted Average" one provides the best outcomes with regards to any of the analysed performance metrics.

3.3. Input and output description

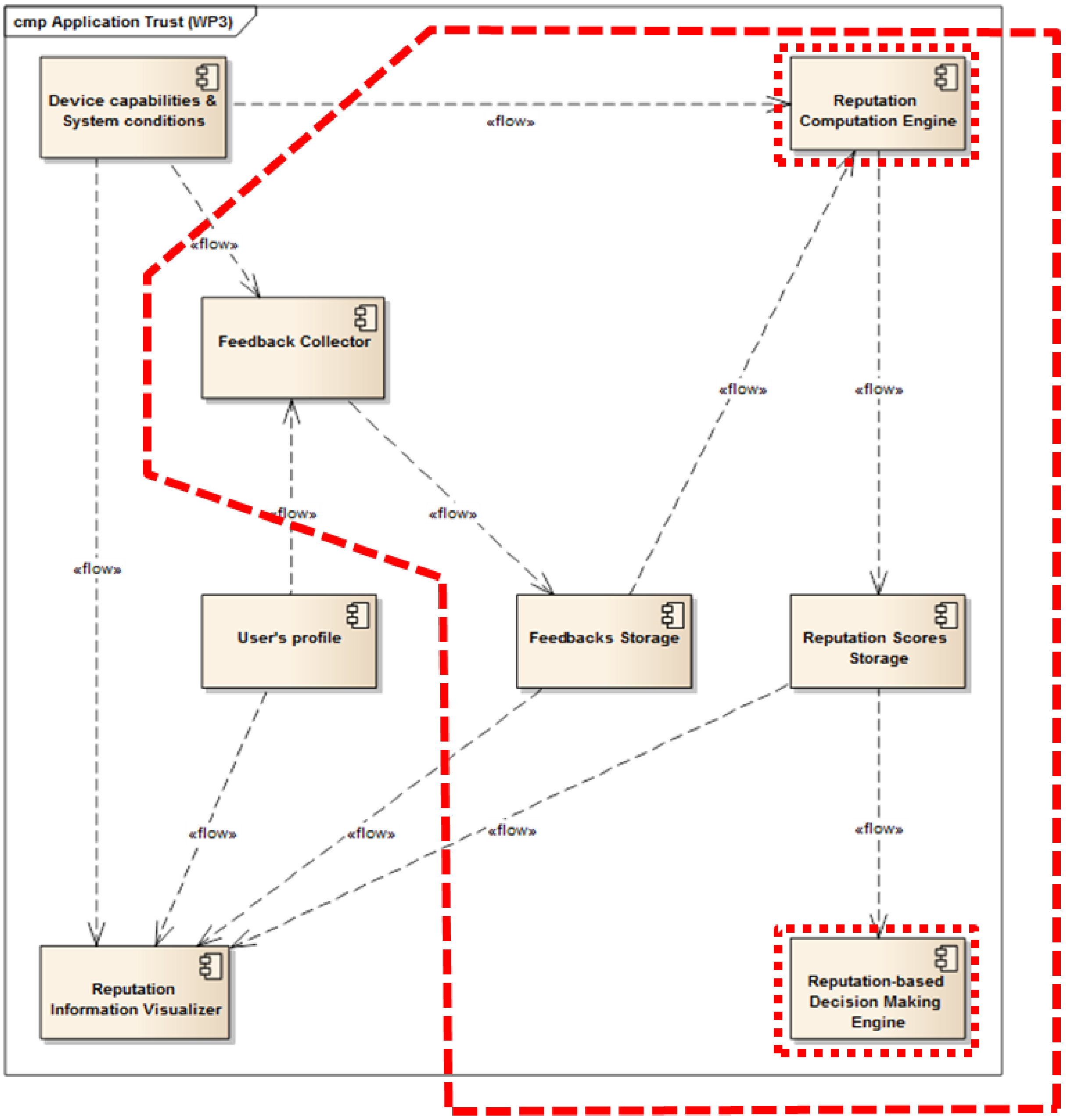

cmp Application Trust (WP3)
Device capabilities & System conditions
Reputation Computation Engine
Feedback Collector
User's profile
Feedbacks Storage
Reputation Scores Storage
Reputation Information Visualizer
Reputation-based Decision Making Engine
«flow»

Figure 14. Trust and reputation enabler's internal components

3.3.1. Internal Implementation

This section shows the updated description of the main operations provided by each one of the internal modules of the trust and reputation management enabler depicted in Figure 14. Note that, in contrast to the modules presented in D3.2 [62], this document focuses only on the submodules actually belonging to the trust and reputation enabler (the ones inside the dashed frame in Figure 14). Furthermore, the submodules presenting the actual research challenges addressed by this enabler, namely, Reputation Computation Engine (see Section **Chyba! Nenašiel sa žiaden zdroj odkazov.**) and Reputation-based Decision Making Engine (see Section **Chyba! Nenašiel sa žiaden zdroj odkazov.**), have been highlighted as well.

3.3.1.1. Feedback collector

The main operation offered by the Feedback collector module, named `collectFeedback()`, has been described in Table 6.

Operation name	`collectFeedback()`	
Operation description	Builds a `<Feedback>` element from the score given by a specific issuer (user) about a concrete subject (HBB-NEXT application), including its timestamp and, optionally, a comment	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	Issuer (user) providing the feedback
`subject`	`<Subject>`	Subject (HBB-NEXT application) to be evaluated by the issuer
`score`	`<Score>`	Feedback score given by the specified issuer about the given subject
`date`	`<Date>`	Timestamp of the feedback provided by the issuer about the subject
`comment`	`<Comment>`	Optional comment associated to the feedback given by the issuer about the subject
Return type	**Return description**	
`<Feedback>`	Returns a `<Feedback>`element containing the score given by the specified issuer about the given subject, as well as a timestamp and, optionally, some opinion message (comment)	

Table 6. Operation: collectFeedback()

3.3.1.2. Feedback storage

The Feedback storage module provides a couple of main operations in order to make use of it. The first one, `storeFeedback()`, has been described in Table 7, while the second one, called `retrieveFeedbackBundle()`, can be found in Table 8.

Operation name	`storeFeedback()`	
Operation description	Stores a given feedback within the Feedbacks storage module in such a way that it will be easily retrievable when needed afterwards	
Parameter name	**Parameter type**	**Parameter description**
`feedback`	`<Feedback>`	Feedback to be stored within the Feedbacks storage module
Return type	**Return description**	
`boolean`	Returns 'true' if the storage of the provided `<Feedback>` element was successful and 'false', otherwise	

Table 7. Operation: storeFeedback()

Operation name	`retrieveFeedbackBundle()`	
Operation description	Retrieves all the feedbacks provided to and stored in the Feedbacks Storage module, regarding a specified subject (HBB-NEXT application)	
Parameter name	**Parameter type**	**Parameter description**
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose feedbacks are to be retrieved
Return type	**Return description**	
`<FeedbackBundle>`	Returns a `<FeedbackBundle>` element containing all the `<Feedback>` elements which, in turn, represent each of the feedbacks provided regarding the specified subject	

Table 8. Operation: retrieveFeedbackBundle()

In order to enable the "Limited Weighted Average" reputation computation engine (Section 0), a slight modification of the operation `retrieveFeedbackBundle()` was required, retrieving a given maximum number of `<Feedback>` elements, as shown in Table 9.

Operation name	`retrieveFeedbackBundle()`
Operation description	Retrieves the feedbacks provided to and stored in the Feedbacks Storage module, regarding a specified subject (HBB-

	NEXT application), up to a given maximum	
Parameter name	**Parameter type**	**Parameter description**
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose feedbacks are to be retrieved
`maxFeedbacks`	`int`	Maximum number of retrieved `<Feedback>` elements
Return type	**Return description**	
`<FeedbackBundle>`	Returns a `<FeedbackBundle>` element containing the `<Feedback>` elements which, in turn, represent each of the feedbacks provided regarding the specified subject, up to a given maximum	

Table 9. Operation: retrieveFeedbackBundle()

3.3.1.3. Reputation computation engine

The main operation of the Reputation computation engine module, which in turn is one of the key operations of the trust and reputation management enabler, has been described in Table 10 and it is called `computeReputation()`.

Operation name	`computeReputation()`	
Operation description	Computes a reputation score for a given issuer, regarding a specified subject (HBB-NEXT application) based on the set of feedbacks provided by previous users regarding that specific subject	
Parameter name	**Parameter type**	**Parameter description**
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose reputation is to be computed
`issuer`	`<Issuer>`	Issuer (user) requesting the computation of the reputation score
`feedbackBundle`	`<FeedbackBundle>`	Set of feedbacks that previous users provided regarding the specified subject
Return type	**Return description**	
`<Reputation>`	Returns a `<Reputation>` element containing the computed reputation score for the specified subject, as well as a timestamp and, optionally, the set of feedbacks provided by previous users regarding that specific subject	

Table 10: Operation: computeReputation()

3.3.1.4. Reputation scores storage

This internal module, Reputation scores storage, offers three main operations to the rest of internal modules within the trust and reputation management enabler, namely: i) `storeReputation()`, described in Table 11, ii) `retrieveReputationBundle()`, shown in Table 12, and iii) `retrieveReputation()`, explained in Table 13.

Operation name	`storeReputation()`	
Operation description	Stores a given reputation within the Reputation scores storage module in such a way that it will be easily retrievable when needed afterwards	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	Issuer (user) requesting the storage of the reputation score
`reputation`	`<Reputation>`	Reputation to be stored within the Reputation scores storage module
Return type	**Return description**	
`boolean`	Returns ‘true’ if the storage of the provided `<Reputation>` element was successful and ‘false’, otherwise	

Table 11. Operation: storeReputation()

Operation name	`retrieveReputationBundle()`	
Operation description	Retrieves each reputation stored in the Reputation scores storage module, regarding each of the specified subjects (HBB-NEXT applications), associated to a given issuer	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	Issuer (user) requesting the retrieval of the reputation bundle
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose reputation is to be retrieved
⋮	⋮	⋮
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose reputation is to be retrieved
Return type	**Return description**	

`<ReputationBundle>`	Returns a `<ReputationBundle>` element containing the `<Reputation>` elements which, in turn, represent each of the reputations stored regarding each of the specified subjects, associated to the given issuer

Table 12: Operation: retrieveReputationBundle()

Operation name	`retrieveReputation()`	
Operation description	Retrieves the reputation stored in the Reputation scores storage module, regarding the specified subject (HBB-NEXT application), associated to the given issuer	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	Issuer (user) requesting the retrieval of the reputation element
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose reputation is to be retrieved
Return type	**Return description**	
`<Reputation>`	Returns a `<Reputation>` element which represents the reputations stored in the Reputation scores storage module regarding the specified subject, associated to the given issuer	

Table 13. Operation: retrieveReputation()

3.3.1.5. Reputation-based decision making engine

The Reputation-based decision making engine module offers only one main operation named `allowAccessToIdentityAttributes()`, whose description can be observed in Table 14.

Operation name	`allowAccessToIdentityAttributes()`	
Operation description	Determines whether to allow or not a specified subject to have access to (some of) the identity attributes of a specified issuer, based on the reputation of the former from the perspective of the latter	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	End user who wants to consume the specified subject and whose identity attributes need to be retrieved

`subject`	`<Subject>`	Subject (HBB-NEXT application) requesting access to certain identity attributes of the specified issuer
Return type	**Return description**	
`{<Permit>|<Deny>}`	Returns either a `<Permit>` element indicating that the specified subject is trustworthy enough to have access to the identity attributes of the specified issuer, or `<Deny>` otherwise	

Table 14. Operation: allowAccessToIdentityAttributes()

3.3.2. External interfaces

This section shows the description of the operations exposed by the trust and reputation enabler (named “Reputation Manager” in the implementation) towards any other external module, enabler or component of the HBB-NEXT architecture.

These operations are:

- `requestReputation()`, with similar arguments and output as the operation `retrieveReputation()` (see Table 13) exposed by the Reputation scores storage
- `provideFeedback()`, with similar arguments and output as the operation `storeFeedback()` (see Table 7) exposed by the Feedback storage
- `requestFeedbackBundle()`, with similar arguments and output as the operations `retrieveFeedbackBundle()` (see Table 8 and Table 9) exposed by the Feedback storage
- `allowAccessToIdentityAttributes()`, with similar arguments and output as the operation `allowAccessToIdentityAttributes()` (see Table 14) exposed by the Reputation-based decision making engine

Furthermore, the two following operations are also part of the Reputation Manager implemented as part of the Trust and reputation enabler: `computeReputation()` (see Table 15) and `updateReputation()` (see Table 16).

Operation name	`computeReputation()`	
Operation description	Computes a reputation score for a given issuer, regarding a specified subject (HBB-NEXT application). The concrete reputation computation engine applied will be selected dynamically depending on the capabilities of the device used to perform the computation of the reputation score	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	Issuer (user) requesting the computation of the reputation score
`subject`	`<Subject>`	Subject (HBB-NEXT application) whose reputation is to be computed
`deviceCapabilities`	`<DeviceCapabilities>`	Capabilities of the device used to compute the reputation score of the specified subject
`systemConditions`	`<SystemConditions>`	Current system conditions, used to select the most suitable reputation computation engine
Return type	**Return description**	
`<Reputation>`	Returns a `<Reputation>` element containing the computed reputation score for the specified subject, as well as a timestamp and, optionally, the set of feedbacks provided by previous users regarding that specific subject	

Table 15. Operation: computeReputation()

Operation name	`updateReputation()`	
Operation description	Updates a given reputation within the Reputation scores storage module in such a way that it will be easily retrievable when needed afterwards	
Parameter name	**Parameter type**	**Parameter description**
`issuer`	`<Issuer>`	Issuer (user) requesting the storage of the reputation score
`reputation`	`<Reputation>`	Reputation to be stored within the Reputation scores storage module
Return type	**Return description**	
`boolean`	Returns ‘true’ if the storage of the provided `<Reputation>` element was successful and ‘false’, otherwise	

Table 16. Operation: updateReputation()

3.4. Future research roadmap

Figure 15 shows the roadmap towards MS8 regarding the trust and reputation enabler, focused on four main topics, namely: i) the integration with the IdM enabler, ii) the development of a mechanism to dynamically select the most appropriate reputation computation engine, based on the current system conditions, iii) the development of a solution to preserve the privacy of users' feedback and iv) a mechanism to automatically retrieve users' attributes based on the reputation of the HBB-NEXT application requesting those attributes.

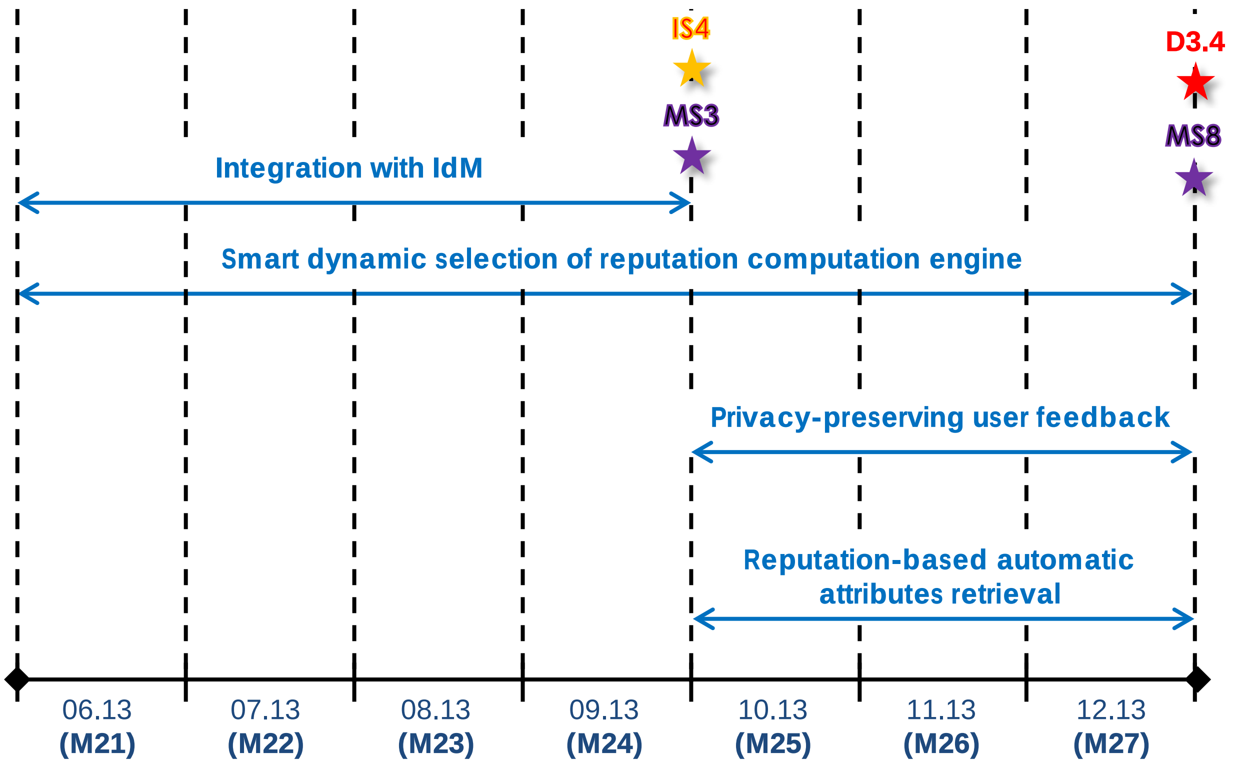

IS4
MS3
D3.4
MS8
Integration with IdM
Smart dynamic selection of reputation computation engine
Privacy-preserving user feedback
Reputation-based automatic attributes retrieval
06.13 (M21)
07.13 (M22)
08.13 (M23)
09.13 (M24)
10.13 (M25)
11.13 (M26)
12.13 (M27)

Figure 15. Trust and reputation enabler: roadmap towards MS8

4. Identity Management Module

This chapter describes one main part of the delivery of WP3: The Identity Management (IdM) module. The module contains all users, basic information about them, security credentials, and connections between users, devices, and contexts.

IdM will provide an API to access these parameters easily, and in a future stage it is possible to add some interpretation to them as well to enable services building logic upon them.

The base of this section is section 4 in the HBB-NEXT Deliverable D3.2. The enabler “Identity Management” is introduced in Section 1.2 in the HBB-NEXT Deliverable D3.1. It is noteworthy that D3.1 does not make distinctions between some functions that have become part of the security manager (see section 6) over time, e.g., authentication.

The IdM module contains all HBB-NEXT users, devices, and the relation between them. Relations are mapped in “contexts” (temporary and independent relations of users and devices as enabler for dynamic adaptive personalization amongst other features). They represent groups of users that consume content/interact together with their HBB-NEXT devices. Contexts can exist over a long period of time – even without any users currently using the system. A distinction is made between defined contexts (“usual contexts”) and contexts in which something is happening at the current moment (“active contexts”). The amount of users and devices in a context is not limited. A context cannot consist of users **or** devices, but always a user **and** device **relation**.

IdM is not only a database that can manage users, contexts, and devices, but it also adds intelligence. At the moment, it will take the input of the multi-modal interface and pre-process it for interpretation by the security manager.

The current work beyond the state of the art, which has been summarized in D3.1, are the definition and implementation of a context model and the way relations are connected. All of this core IdM implementation has been made available in a real RESTful API design (per Fielding’s definition), incl. the implementation of a major part of it. Moreover, IdM will do more than only gather information. It will process them, e.g., “raw” MMI data is being normalized. Moreover, the time frame on which you are usually active might have impact as well on the accuracy of the provided values (not part of the implementation).

It does not only "store/manage" ID information, but also "interpret/enhance" them. It is not only a processor but enabler.

The following points from the gap analysis of D3.1 have been addressed so far:

- Gap 1 “user relations”: A solution has been designed. User relations are mapped through the context. Direct relations between users are possible, but not part of the current HBB-NEXT model.
- Gap 2 “’smart’ enabler”: A solution has been designed. This part is still under implementation (e.g. integration of normalization), but finalization until the end of the project is secured.
- Gap 3 “user profile improvements”: A solution has been designed. The profile management has an API that allows user profiles as well as pre-configured service profile mappings. The implementation will be done that a basic use case with the personalization engine (PE) in WP5 can be shown.
- Gap 4 “multi-screen enabler”: A solution has been designed. A use case is under implementation, the API usage for this kind of scenarios is provided within this section.

Gap 5 is security related and thus shifted to the scope of the Security Manager (SecM).

This section will describe the IdM data model in detail, show how the enabler has been designed, provide details of the RESTful API, and provide detailed use cases for the particular app requirements.

At the current state, the IdM design is almost complete and the implementation is at a stage where integration has been already performed (e.g. SecM, MMI). The work until M27 will be minor design adjustments where needed as well as implementation and integration of the prototype for demonstrating the application showcases.

4.1. Data model

This section contains the data model.

The data model field obligation is indicated with ‘M/O’ - meaning ‘mandatory/optional’. The amount of objects is indicated in the occurrences column with ‘1/n’ - meaning ‘unique/multiple’.

The JSON type ‘boolean’ summarizes true/false as permitted values for the object.

The JSON type ‘date’ specifies a string as permitted value for the object in the format YYYY-MM-DD acc. ISO 8601, where YYYY equals the four-digit year, MM equals the two-digit month (01=January, etc.), and DD equals the two-digit day of month (01 through 31).

The first value in an array is the default value. Other values shall be used or can be selected according to particular use cases.

4.1.1. User

The user data model is shown in Table 17. The JSON type refers to the specification from www.json.org. It is for example implicit that the password shall not be provided as ‘bare string’ but rather MD5 hashed (or any format that is required by apps) and considered secure.

Level 1	Level 2	Level 3	JSON Type	JSON Object	Obligation	Ocurrences
Node			object		M	1
Identifier (unique)						
	ID		number	id	M	1
	Alias		array	alias	O	n
	Username		string	username	O	1
	UUID		string	uuid	M	1
MMI			object	mmi	M	1
	Voice value		number (digits)	voice	M	1
	Face value		number (digits)	face	M	1
Meta			object	meta	O	1
	Description		string	description	O	1
	Success indicator		boolean	success	O	1
	User who created the entry		string	created_by	O	1
	Date of creation		datetime	created_at	O	1
	Date of last update		datetime	updated_at	O	1
	API version		string	version	O	1
Personal information			object	info	O	1
	Gender		string	gender	O	1
	First Name		*string*	*first_name*	O	1
	Last Name		*string*	*last_name*	O	1
	Date of birth		date	birthday	O	1
Contact means			object	contact	O	1
	E-mail addresses		array	email	O	n
	Phone number		array	phone	O	n
Credentials			object	credentials	O	1
	Password		string	password	O	1
	PIN		number (digits)	pin	O	1
Links			object	links	O	1
	Device		array	devices	O	n
		Device ID	number	id	M	1
		Name	string	name	M	1
		API link	string	href	M	1
	User (incl. weight: family, friend, implicit/explicit, etc.)		*array*	*users*	O	n
	Context		array	contexts	O	n
		Context ID	number	id	M	1
		Display name	string	display_name	M	1
		API link	string	href	M	1
	Profile		*array*	*profile*	O	1

API version: 3
Date: 17.5.13

Table 17. User data model

The data model may be extended for future API versions.

The 'link' type is a reference pointing to the API ID for the respective element. In a later version of the API more than just bare objects/arrays may be provided. The IdM enabler is aimed provide more intelligence for providing related information. Links between elements may be weighted, i.e., they are not only directed connections between two elements (whereas 'device' is the origin), but may have a label (weight) that specifies the connection further. This can be used to parameterize connections between users further (not only *that* I know another user, but *how*).

Table 17 shows only the high level JSON type. Due to the fact that this specification is not final the content is subject to changes.

A JSON sample of the data model is shown below:

```
{
    "id": 15,
    "alias": "testUser",
    "username": "testUser",
    "uuid": "7bcfc374-9770-413a-b036-52d82baa4163",
    "mmi": {
        "voice": "20",
        "face": "10"
    },
    "meta": {
        "description": "n/a",
        "success": true,
        "created_by": "n/a",
        "created_at": "2012-10-28T21:14:57Z",
        "updated_at": "2013-05-10T10:02:22Z",
        "version": "v3"
    },
    "credentials": {
        "password": "",
        "pin": "0000"
    },
    "info": {
        "gender": "false",
        "birthday": "1900-01-01"
    },
    "contact": {
        "email": "test@example.com"
    },
    "links": {
        "contexts": [
            {
                "id": 5,
                "name": "testContext",
                "href": "http://hbbnext-idm.dev/api/v3/contexts/5"
            }
        ],
        "devices": [
            {
                "id": 9,
                "display_name": "testDevice",
                "href": "http://hbbnext-idm.dev/api/v3/devices/9"
            }
        ]
    }
}
```

It could be the body of a response to the following HTTP request:

```
GET /api/v3/users/15.json HTTP/1.1
User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0
OpenSSL/0.9.8r zlib/1.2.5
Host: hbbnext-idm.dev
Accept: */*
```

4.1.2. Device

The device data model is shown in Table 18. The JSON type refers to the specification from www.json.org.

Level 1	Level 2	Level 3	JSON Type	JSON Object (preliminary)	Obligation	Ocurrences
Node			object		M	1
Identifier (unique)			object		M	1
	ID		string	id	M	1
	Alias		string	alias	O	n
	Display name		string	display_name	O	1
	address		string	address	M	1
	UUID		string	uuid	M	1
Meta			object	meta	O	1
	Description		string	description	O	1
	Success indicator		boolean	success	O	1
	User who created the entry		string	created_by	O	1
	Date of creation		datetime	created_at	O	1
	Date of last update		datetime	updated_at	O	1
	API version		string	version	O	1
Links			object	links	O	1
	Users		array	users	O	n
		User ID	number	id	M	1
		Username	string	username	M	1
		API link	string	href	M	1
	Context		array	contexts	O	n
		Context ID	number	id	M	1
		Display name	string	display_name	M	1
		API link	string	href	M	1

API version:	3
Date:	17.5.13

Table 18. Device data model

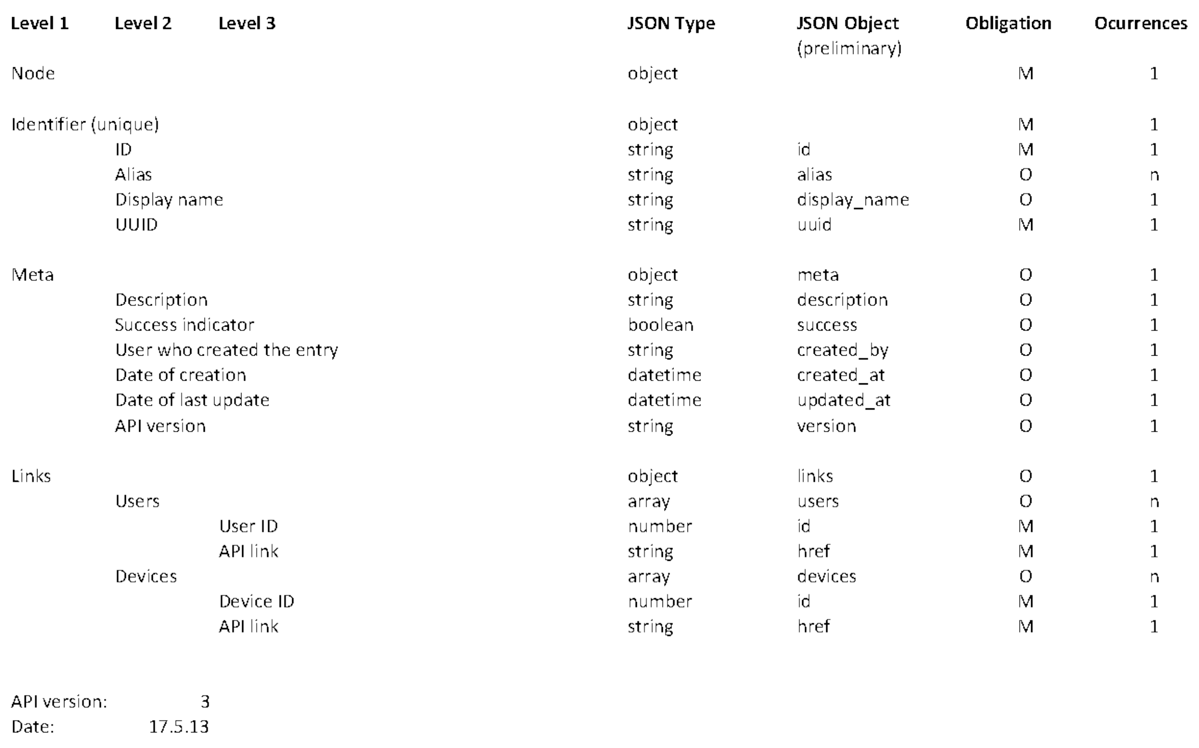
The data model may be extended for future API versions.

The properties element can contain device specific sub-elements. There is currently no need to define that, as it is used per service and data is transparent to the API. This statement may however be subject to future change.

The ‘link’ type is a reference pointing to the API ID for the respective element. Links between elements may be weighted, i.e., they are not only directed connections between two elements (whereas ‘device’ is the origin), but may have a label (weight) that specifies the connection further. This can be used to parameterize the relation of e.g. a user to a device (e.g. watcher, owner, user, etc.). Further logic can process and use this information.

4.1.3. Context

The context data model is shown in Table 19. The JSON type refers to the specification from www.json.org.



Level 1	Level 2	Level 3	JSON Type	JSON Object (preliminary)	Obligation	Ocurrences
Node			object		M	1
Identifier (unique)			object		M	1
	ID		string	id	M	1
	Alias		string	alias	O	n
	Display name		string	display_name	O	1
	UUID		string	uuid	M	1
Meta			object	meta	O	1
	Description		string	description	O	1
	Success indicator		boolean	success	O	1
	User who created the entry		string	created_by	O	1
	Date of creation		datetime	created_at	O	1
	Date of last update		datetime	updated_at	O	1
	API version		string	version	O	1
Links			object	links	O	1
	Users		array	users	O	n
		User ID	number	id	M	1
		API link	string	href	M	1
	Devices		array	devices	O	n
		Device ID	number	id	M	1
		API link	string	href	M	1

API version:	3
Date:	17.5.13

Table 19. Context data model

The data model may be extended for future API versions.

The ‘link’ type is a reference pointing to the API ID for the respective element.

Contexts play a major role in the implementation of the identity management. The IdM uses the context element to map the usual and active context of a user. While the context assignment in WP3 is rather static and only its activity changes, the context assignment in scope of WP5 (see D5.2 [16]) is dynamic.

The “usual context” represents the context (defined in D3.1 [18]), in which a user is usually present, i.e., has been present before. It is besides other factors relevant for defining the sub-set for multi-modal identification.

The “active context” represents the current context of a user, where he is presently active in. It is used for interaction.

“Context” is used to group users and devices. They can, however, still have direct relations.

4.2. Design

The IdM is designed to be accessed mainly by the Security Manager. It can as well be accessed by all the enablers that have a valid security token. Since the profile management is tightly coupled with the IdM, it is one of the enablers that can access it by default. The same is true to the personalization engine, which is going to be implemented as part of the work in WP5.

The module consists of front-end, back-end, and database layers.

The front-end is an HTTP web service in the form of a RESTful API. The back-end contains the module logic and communicates with the data base layers.

The data model shows the three main domains:

- User
- Device
- Context

Typically, users are permanently related to each other and to their respective devices.

The context maps a temporary relation between users (that may or may not be permanently related, but most likely are) and between devices.

Relations may have a “weight”, i.e., roles that further define the relation, but this relation is not mandatory. Users can administer other users (e.g. TV owner can create user accounts) and also devices (e.g. device owner can configure the device). Users can also use devices – with restricted permissions.

Figure 16 shows the relations between the domains in both identity and profile management.

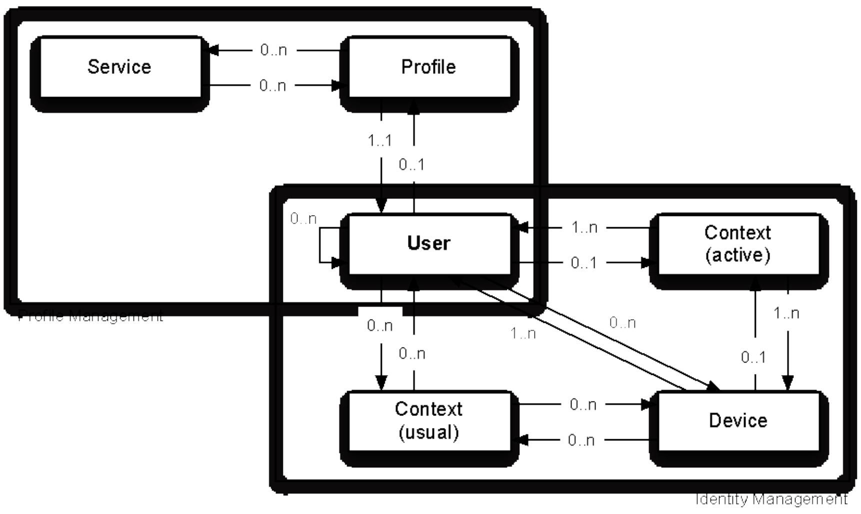


Figure 16. Relations between identity and profile management domains

A user may have relations to many other users.

A user can have many devices. Devices must belong to one or many users and their relation is weighted.

A user can only be active in a single context. An active context can have one or many users attached and its relation may be weighted towards the controlling user.

A user can be part of many usual contexts. A usual context can have users attached.

A device can only be active in a single context. An active context can have one or many devices attached and its relation may be weighted towards the controlling device.

A device can be part of many usual contexts. A usual context can have devices attached.

A user can have only one profile. A profile is always assigned to a particular user. Services may be assigned to profiles.

Relations of profile and service are covered in section 5.

4.3. Interface

4.3.1. Internal Implementation

The identity management module has been implemented according to the design that is described in section 0.

4.3.2. API

The basic functions of the identity management component are described in D6.1.1, section 6.3.4 [19]. The mentioned functions in this deliverable are not elaborated and mapped to the API.

Note: In case no parameters are added, this shall be indicated by inserting 'n/a' (not applicable) into the table.

Note: Data types for the parameters are stated in section 0 covering the data model.

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete user in the system**		
Function:	Retrieve user		
Method:	GET	**Resource:**	/users/*{userid}*
Parameters:			
userid	The id of the user that shall be retrieved.		
BODY	The body of the message shall contain basic user parameters acc. [ref. this doc, sec. 4.1]		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete user in the system**		
Function:	Add user		
Method:	POST	**Resource:**	/users
Parameters:			
BODY	The body of the message may contain user parameters acc. [ref. this doc, sec. 4.1]. These parameters are provisioned to the created user.		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete user in the system**		
Function:	Modify user		
Method:	PUT	**Resource:**	/users/*{userid}*
Parameters:			
userid	The id of the user that shall be modified.		

BODY	The body of the message shall contain user parameters acc. [ref. this doc, sec. 4.1]. These parameters are provisioned to the user that shall be updated. They will override existing parameters.

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete user in the system**		
Function:	Delete user		
Method:	DELETE	**Resource:**	/users/*{userid}*
Parameters:			
userid	The id of the user that shall be deleted.		

Function (Ref. D6.1.1):	**Add, modify, delete user from a context**		
Function:	Create context		
Method:	POST	**Resource:**	/contexts
Parameters:			
n/a	n/a		

Function (Ref. D6.1.1):	**Add, modify, delete user from a context**		
Function:	Add user to context		
Method:	POST	**Resource:**	/contexts/*{contextid}*/users?id=*{userid}*
Parameters:			
contextid	The context that is being modified.		
userid	The ID of the user to be added.		

Note: This will not be implemented for the prototype.

Function (Ref. D6.1.1):	**Add, modify, delete user from a context**		
Function:	Modify user in context (necessity to be discussed in further study)		
Method:	PUT	**Resource:**	/contexts/*{contextid}*/users/*{userid}*
Parameters:			
contextid	The context that is being modified.		
userid	The ID of the user to be modified.		

Note: This will not be implemented for the prototype.

Function (Ref. D6.1.1):	**Add, modify, delete user from a context**		
Function:	Delete user from context		
Method:	DELETE	**Resource:**	/contexts/{contextid}/users/{userid}
Parameters:			

contextid	The context that is being modified.
userid	The ID of the user to be deleted.

Note: This will not be implemented for the prototype.

Function (Ref. D6.1.1):	**Add, modify, delete user from a context**		
Function:	Delete context		
Method:	DELETE	**Resource:**	/contexts/{contextid}
Parameters:			
contextid	The context that is being deleted.		

Function (Ref. D6.1.1):	**Retrieve users active in a context (at the present moment)**		
Function:	see above		
Method:	GET	**Resource:**	/contexts/{contextid}/active/users
Parameters:			
contextid	The context that is being used.		

Function (Ref. D6.1.1):	**Retrieve users relevant for a context (usually active there)**		
Function:	see above		
Method:	GET	**Resource:**	/contexts/*{contextid}*
Parameters:			
contextid	The context that is being used.		
links:users	All users that are assigned to the context.		

Function (Ref. D6.1.1):	**Retrieve devices active in a context (at the present moment)**		
Function:	see above		
Method:	GET	**Resource:**	/contexts/*{contextid}*/active/devices
Parameters:			
contextid	The context that is being used.		

Function (Ref. D6.1.1):	**Retrieve devices relevant for a context (usually active there)**		
Function:	see above		
Method:	GET	**Resource:**	/contexts/*{contextid}*
Parameters:			
contextid	The context that is being used.		
links:devices	All users that are assigned to the context.		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete device in the system**		
Function:	Retrieve device		
Method:	GET	**Resource:**	/devices/{deviceid}
Parameters:			
deviceid	The device id of the device that shall be retrieved.		
BODY	The body of the message shall contain basic device parameters acc. [ref. this doc, sec.]		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete device in the system**		
Function:	Add device		
Method:	POST	**Resource:**	/devices
Parameters:			
BODY	The body of the message shall contain device user parameters acc. [ref. this doc, sec.]		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete device in the system**		
Function:	Modify device		
Method:	PUT	**Resource:**	/devices/{deviceid}
Parameters:			
deviceid	The device id of the device that shall be retrieved.		
BODY	The body of the message shall contain basic device parameters acc. [ref. this doc, sec.]		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete device in the system**		
Function:	Delete device		
Method:	DELETE	**Resource:**	/devices/{deviceid}
Parameters:			
deviceid	The device id of the device that shall be retrieved.		

Function (Ref. D6.1.1):	**Add, modify, delete device from a context**		
Function:	Add device to context		
Method:	POST	**Resource:**	/contexts/*{contextid}*/devices?id=*{deviceid}*
Parameters:			
contextid	The context that is being modified.		
deviceid	The ID of the device to be added.		

Note: This will not be implemented for the prototype.

Function (Ref. D6.1.1):	**Add, modify, delete device from a context**		
Function:	Modify device in context (necessity to be discussed in further study)		
Method:	PUT	**Resource:**	/contexts/{contextid}/devices/{deviceid}
Parameters:			
contextid	The context that is being modified.		
deviceid	The ID of the device to be added.		

Note: This will not be implemented for the prototype.

Function (Ref. D6.1.1):	**Add, modify, delete device from a context**		
Function:	Delete device from context		
Method:	DELETE	**Resource:**	/contexts/{contextid}/devices/{deviceid}
Parameters:			
contextid	The context that is being modified.		
deviceid	The ID of the device to be added.		

Note: This will not be implemented for the prototype.

Function (Ref. D6.1.1):	**Retrieve devices of a user**		
Function:	see above		
Method:	GET	**Resource:**	/users/*{userid}*
Parameters:			
userid	The user that is being used.		
links:devices	Devices of the user		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete relation between a user (relation, value)**		
Function:	Retrieve relation between users		
Method:	GET	**Resource:**	/users/*{userid}*/users
Parameters:			
n/a	n/a		

Note: This will not be available for the prototype (neither web interface nor API).

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete relation between a user (relation, value)**		
Function:	Add/modify relation between users		
Method:	PUT	**Resource:**	/users/*{userid1}*/users?id=*{userid1}*&bi=*{bi}*&w=*{w}*
Parameters:			
userid1	ID of user to be modified		
userid2	ID of user relation to be added		
bi	Indicator whether relation shall be added bilateral (default: no)		
w	weight of the relation		

Note: This will not be available for the prototype (neither web interface nor API).

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete relation between a user (relation, value)**		
Function:	Delete relation between users		
Method:	DELETE	**Resource:**	/users/{userid1}/users?id={userid1}&bi={bi}
Parameters:			
userid1	ID of user to be modified		
userid2	ID of user relation to be deleted		
bi	Indicator whether relation shall be added bilateral (default: no)		

Note: This will not be available for the prototype (neither web interface nor API).

Function (description):	**Supporting function**		
Function:	API description: Retrieve available resources		
Method:	GET	**Resource:**	/resources
Parameters:			
n/a	n/a		

The resource that is mentioned in the API description is adhered to the root URL, for example:

- Root URL: *https://idm.example.org/api/v1/*
- Full URL for first case: *https://idm.example.org/api/v1/user/a1b2c3*

The response code will provide general information about the success of the request. The response headers carry information relevant to the request and further requests. The response bodies of all requests will contain the complete data sets.

Data can be searched using the search string for the value in the form */search/users?firstName=Jon*. The answer will be provided in the body of the HTTP response message.

Unless specified otherwise, the default content is used in the body. The requested resource might specify the content type in case multiple content types are offered (e.g. */users.json* or */users.xml*).

The following functions (based on D6.1.1 [19]) are not explicitly mapped in the API description above, but rather build with the existing commands:

- Provide multi-modal vectors for users relevant for a context (single modes, all modes)
- Fulfill requirements of security management (e.g. credentials)
- Retrieve, add, modify, delete roles of a user
- Attach, detach user to device (*PUT /users/{userid}?device={deviceid}*)

Further documentation will provide samples how this can be done.

4.4. App team requirements

The scope of this section is to provide an overview of the current implementation of the Identity Management (IdM) within Work Package (WP) 3 of the HBB-NEXT project.

It refers to the list of IdM/Security Manager (SecM) requirements shown in Table 20.

IdM/SecM requirements			
Req Nr	Requirement	App	Note
ISR-F-1	Who are the users in the TNO demo context that have been identified with a certain probability?	2	
ISR-F-2	Provide a callback mechanism if there is any change.	2	IdM will issue a notification towards the notification framework when the status of a user in a context changes.
ISR-F-3	Identify the users that use the TV set usually (implicit for MMI)	6	

IdM/SecM requirements			
Req Nr	Requirement	App	Note
ISR-F-4	Set recognized probability of certain users on a certain device/in a certain context (implicit for MMI)	6	Currently not implemented per context, only user.
ISR-F-5	Get active users	6	
ISR-F-12	Connect a TV to a second screen device	0	Backend only.
ISR-F-13a	Provide "profile info" with all users. (Users shall be able to login from there.)	0	Backend only.
ISR-M-1	How is the step-by-step setup procedure, i.e., how are the initial IdM entries created?	0	
ISR-M-3	How are multiple entries avoided?	0	
ISR-R-1	Accessible fields in DB model ((user)ID, name, password, mail, last login, when account was created)	1	
ISR-R-2	Handling of anonymous users (if user is not logged in, he shall still be able to use personalization)	1	
ISR-R-3	Retrieve login status per device	1	
ISR-R-4	Retrieve all devices a user has ever logged on	1	
ISR-R-5	Retrieve last device a user was logged on	1	Only per context.
ISR-R-6	Ability to have different login status between first and second screen device	1	
ISR-R-8	Activate an account via activation link in mail	1	This step can be implemented on top of IdM, but IdM does not foresee activation for the time being.
ISR-R-10	Offer the functionality to manage different devices and how they are and were connected to each other and users (e.g. connect/disconnect devices)	1	Backend only.
ISR-R-11	Evaluate the prioritization between different application settings? (one wants subtitles turned on, the other doesn't)	1	Covered by personalization engine.
ISR-R-12	MMI will periodically or non-periodically (event/action-based) send modalities percentages (e.g. face:60%, voice:40%) to IdM. IdM should store these values so they can be read from there through defined API for future purpose (e.g. for apps)	0	

IdM/SecM requirements			
Req Nr	Requirement	App	Note
ISR-R-13	After a defined time, the modalities values in IdM should be marked as "expired" (due to situations when a user authenticates himself for some reason and after a while he leaves the room - he cannot be recognized as still present)	0	This will happen automatically if the user is not present anymore. Otherwise SecM takes care of how long authorizations are valid.

Table 20: IdM requirements

The listed requirements are all application requirements that have been introduced by WP6 for the creation of the HBB-NEXT demo apps.

The following sub-sections address these requirements, describe the API interaction, and thus act as guideline for interaction with the IdM API in the context of HBB-NEXT.

The IdM design fulfils all requirements, a majority of requirements is also being implemented in the demo enabler.

4.4.1. ISR-F-1

Requirement

List active users in a certain context.

Status

This requirement is accepted and implemented.

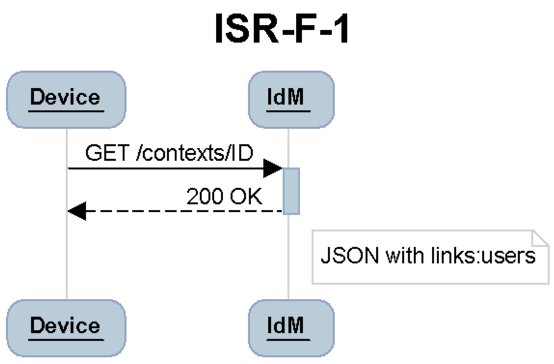
Description

1. Precondition

The following preconditions apply:

- It is assumed that at least one user logged in and the device is aware of the context ID.

2. Sequence Diagram



4.4.2. ISR-F-2

Requirement

Call-back interface for changes in a certain context.

Status

This requirement is accepted. The implementation is pending. The interface will be described within the next sub-section.

Description

The description will be provided in a later version of this document.

4.4.3. ISR-F-3

Requirement

Identify the users that use the TV set usually (users on a device).

Status

This requirement is accepted and implemented.

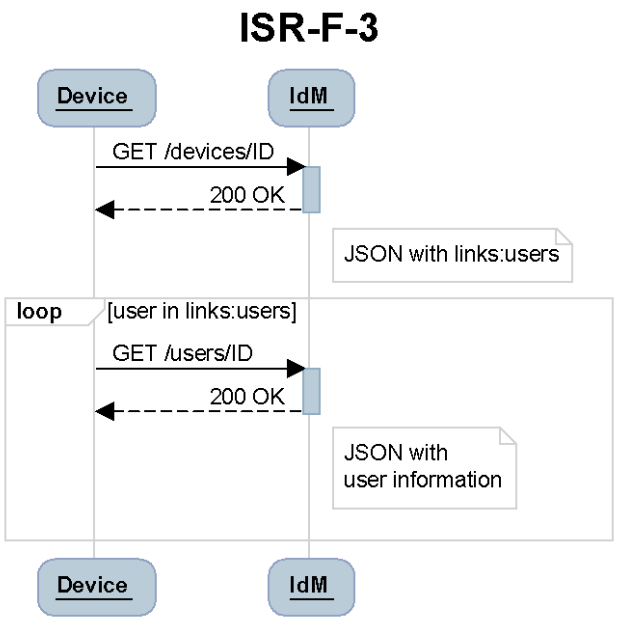
Description

1. Precondition

The following preconditions apply:

- It is assumed that the device knows its ID.

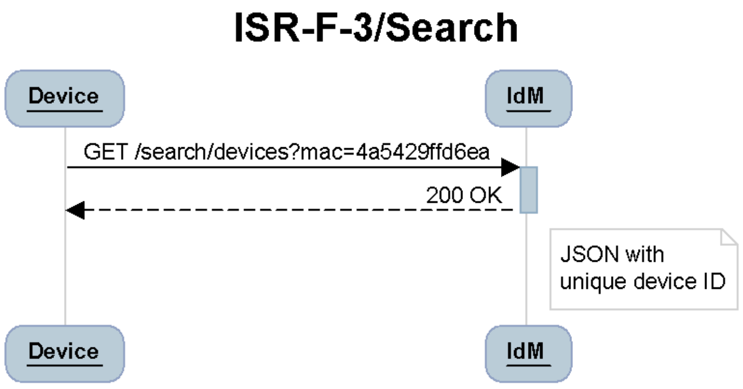
2. Sequence Diagram



Note

If a device does not know its ID, it can easily query it. Since it is created initially with a unique key (see also use case ISR-M-1), a search for this unique key will always result in a single device ID as response.

A search sample is given below:


ISR-F-3/Search
Device
IdM
GET /search/devices?mac=4a5429ffd6ea
200 OK
JSON with unique device ID
Device
IdM

Note: The search interface is not yet implemented.

4.4.4. ISR-F-4

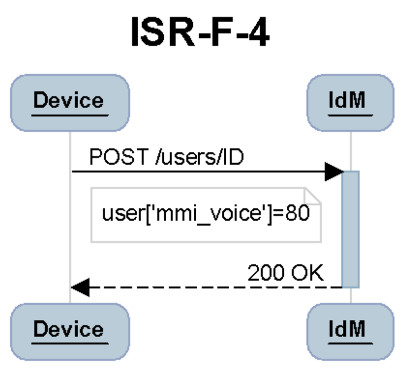
Requirement

Set recognized probability of certain users on a certain device/in a certain context (implicit for MMI).

Status

This requirement is implemented for a global user. It is not yet available per context.

Description



Note

The MMI values need to be passed as x-www-form-urlencoded parameters.

Sample:

- user[mmi_face]=80
- user[mmi_voice]=30

4.4.5. ISR-F-5

Requirement

Get active users.

Status

This requirement is accepted and partially implemented (not via API yet).

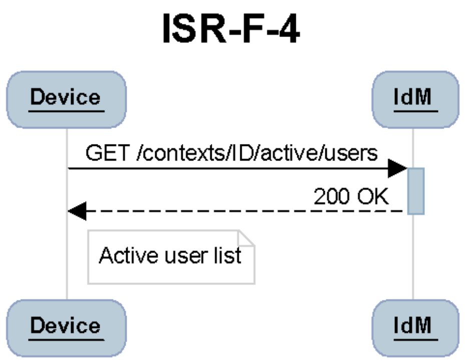
Description

1. Precondition

The following preconditions apply:

- It is assumed that the device knows the context ID.

Sequence Diagram



4.4.6. ISR-F-12

Requirement

Connect two devices, e.g., connect a TV to a second screen tablet.
This requirement connects two active devices in one context used by a single user.

Status

This requirement is accepted and partially implemented (not via API yet).

Description

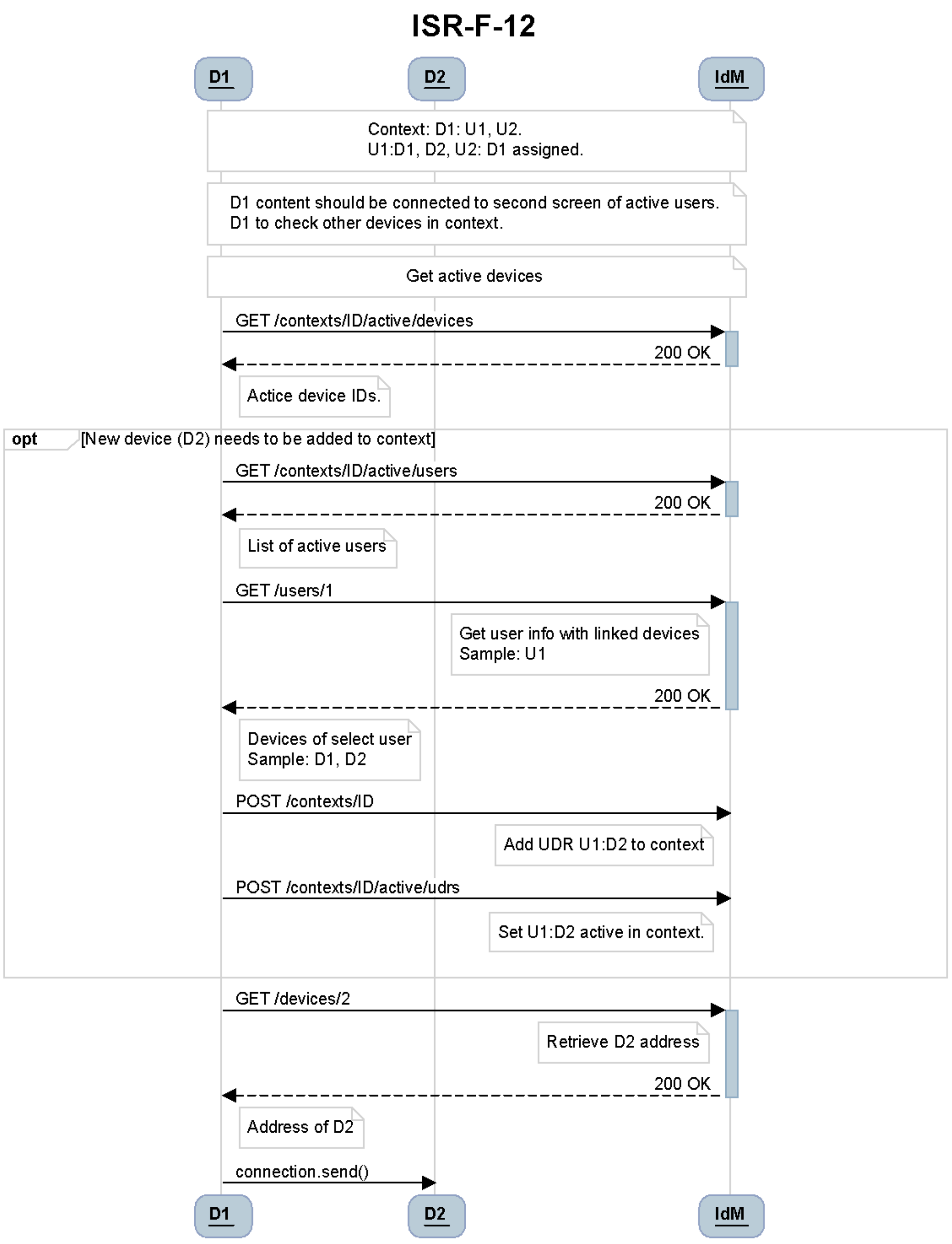
1. Precondition

The following preconditions apply:

- It is assumed that the device knows the context ID and the users
- In the sample, user U1 and user U2 are watching together on the device D1. User U1 plans to add second screen content to the device D2. All users are active in the context C1.

Sequence Diagram

The following sample connects the devices in one context:


ISR-F-12
D1
D2
IdM
Context: D1: U1, U2.
U1:D1, D2, U2: D1 assigned.
D1 content should be connected to second screen of active users.
D1 to check other devices in context.
Get active devices
GET /contexts/ID/active/devices
200 OK
Actice device IDs.
opt
[New device (D2) needs to be added to context]
GET /contexts/ID/active/users
200 OK
List of active users
GET /users/1
Get user info with linked devices
Sample: U1
200 OK
Devices of select user
Sample: D1, D2
POST /contexts/ID
Add UDR U1:D2 to context
POST /contexts/ID/active/udrs
Set U1:D2 active in context.
GET /devices/2
Retrieve D2 address
200 OK
Address of D2
connection.send()
D1
D2
IdM

Note

The following notes apply to the sample:

- User U1 and U2 were linked to the device D1 prior to the start of the showcase
- User U1 was linked to the device D2 prior to the start of the showcase

Note: Some requests require additional parameters to be transmitted.

4.4.7. ISR-F-13a

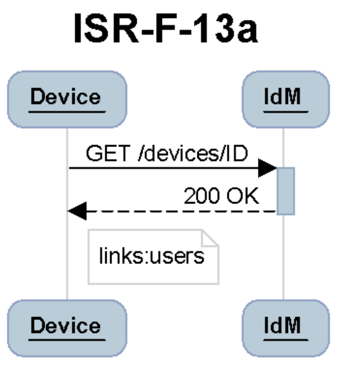
Requirement

Provide info with all users, so a user list can be built and users can login from there.

Status

This requirement is accepted and implemented.

Description



Note

As usual, IdM does not provide a front-end – only means to retrieve the list on the backend.

4.4.8. ISR-M-1

Requirement

Describe a step-by-step setup procedure how the initial IdM entries are created.

Status

This requirement is accepted and partially implemented.

Description

1. Precondition

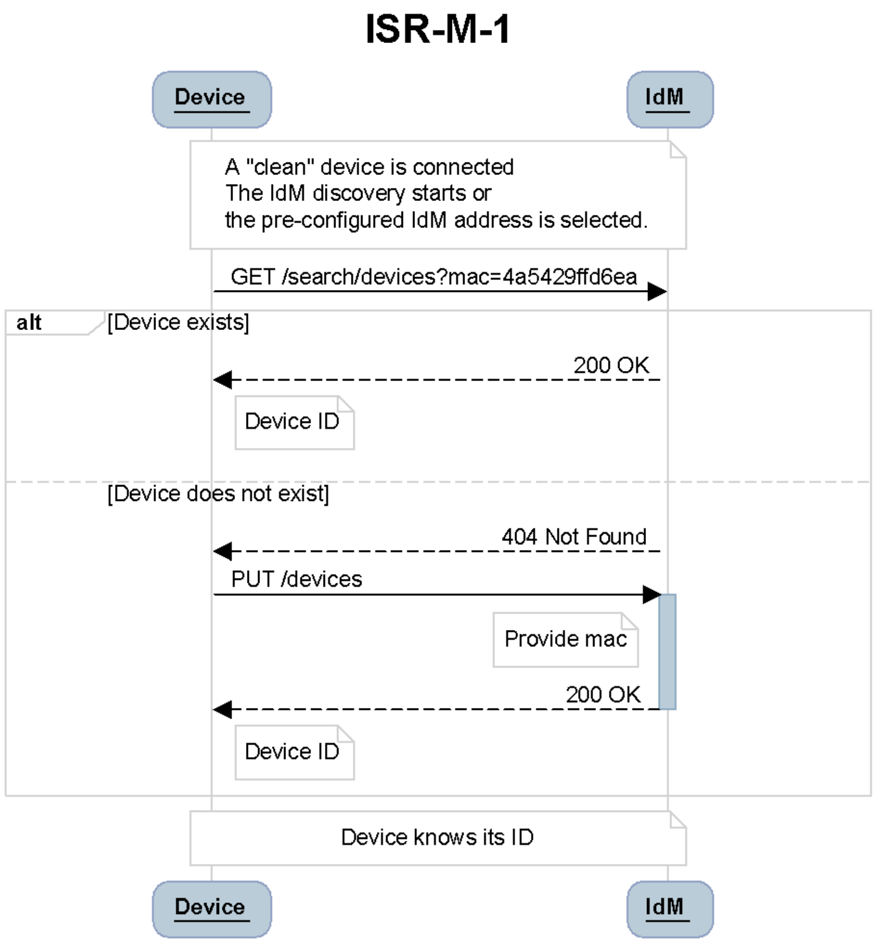
The following preconditions apply:

- It is assumed that neither device nor users are known to the system.
- For auto-provisioning of the device to the IdM, the device must have permission to access the IdM (e.g. token).
- In case the device does not have permission to register itself, a user can register itself first and then register the device in a second step.

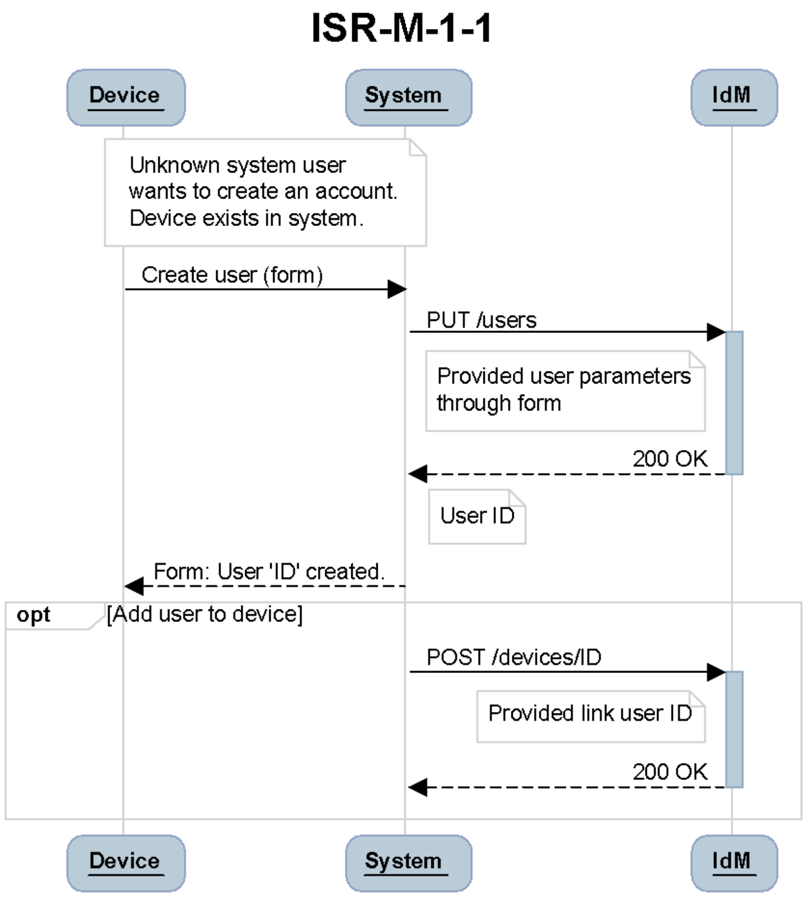
- Provisioning scenarios are very flexible, the following sequence diagram shall provide only a sample scenario.
- On initial creation, the device will get a unique ID assigned within IdM. To be able to have a non-provisioned device check for its IdM device ID, it needs to be able to generate the UID. Alternatively, the network infrastructure can be used as well.
- In the sample below, the MAC address is used as unique ID. It could as well be an IP address (assuming it is unique within the operator network and its assignments are known) or an IMSI/IMEI address for mobile devices. iOS (had) and Android devices have also a unique ID that could be used.

Sequence Diagram

The following diagram shows how to add a new device.



The following diagram shows how a system component (e.g. SecM) can add a new user and optionally link him to the device.



Note

The interfaces are very flexible.

Device linking is not yet implemented on API level (only GUI).

4.4.9. ISR-M-3

Requirement

How are multiple entries avoided?

Note

Certain values have to be unique, otherwise an error response is provided.

The ID is always unique. For certain values (e.g. MAC), a check will be implemented in a future release.

4.4.11. ISR-R-1

Requirement

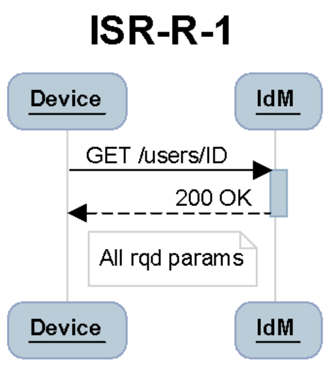
The following user fields shall be accessible:

- User ID
- Name
- Password
- Mail
- Last login
- Date of account creation

Status

All user fields will be exposed via the API. All but the „last login" user fields are implemented.

Description



4.4.12. ISR-R-2

Requirement

Handling of anonymous users (if user is not logged in, he shall still be able to use personalization).

Status

The handling of anonymous users is not considered to be anyhow special. In case an app would like to create a user, use this user without any assignment to a real user and delete it at a later stage, IdM does not impose any restriction.

4.4.14. ISR-R-3

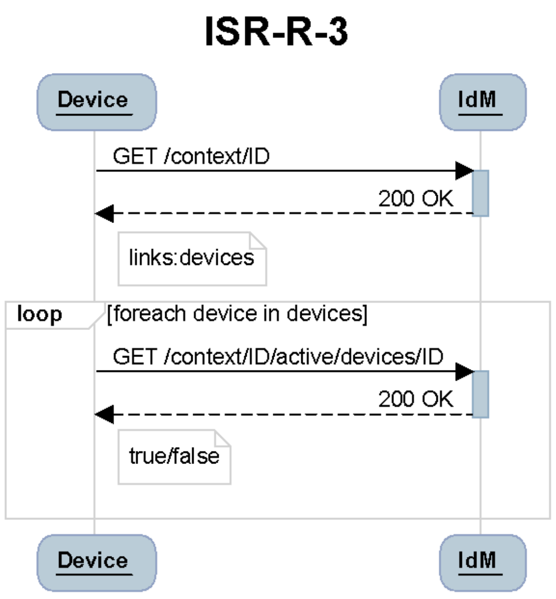
Requirement

Retrieve login status per device.

Status

This requirement is accepted and implemented.

Description



Note

The active-device relation in context indicates both active users as well as active devices. Active is considered logged in via the SecM.

4.4.15. ISR-R-4

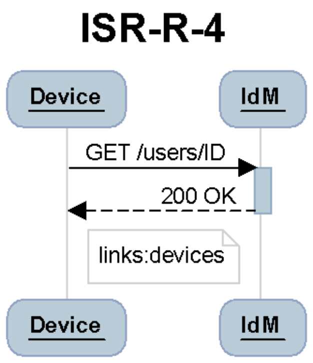
Requirement

Retrieve all devices a user has ever logged on.

Status

This request is accepted and implemented.

Description



Note

All linked devices a user has are considered sufficient for answering this request.

4.4.16. ISR-R-5

Requirement

Retrieve last device a user was logged on

Status

This requirement is accepted and should be implemented in the future.

Note

Once the “last login” timestamp will be provided for a user/device relation (last time the relation has been active) this requirement will be fulfilled.

4.4.17. ISR-R-6

Requirement

Ability to have different login status between first and second screen device.

Status

This requirement is rejected. A user is either active or not active according the current model.

4.4.18. ISR-R-8

Requirement

Activate an account via activation link in mail.

Status

This requirement is rejected.

This step can be implemented on top of IdM, but IdM does not foresee activation for the time being.

4.4.19. ISR-R-10

Requirement

Offer the functionality to manage different devices and how they are and were connected to each other and users (e.g. connect/disconnect devices).

Status

This requirement is implemented for the web front-end and available in the API description. The actual implementation will depend on the available resources.

4.4.20. ISR-R-11

Requirement

Evaluate the prioritization between different application settings.

Status

This requirement is rejected.

The personalization engine should cover this requirement.

4.4.21. ISR-R-12

Requirement

The MMI will periodically or non-periodically (event/action-based) send modalities percentages (e.g. face:60%, voice:40%) to IdM. IdM should store these values so they can be read from there through defined API for future purpose (e.g. for apps)

Status

This request is accepted and implemented. It is similar to ISR-F-3.

4.4.22. ISR-R-13

Requirement

After a defined time, the modalities values in IdM should be marked as "expired" (due to situations when a user authenticates himself for some reason and after a while he leaves the room - he cannot be recognized as still present)

Status

This requirement should not require any additional work from IdM. It will happen automatically if the user is not present anymore. Otherwise SecM takes care of how long authorizations are valid.

5. Profile Management Module

This chapter describes the profile management (PM) module.

The profile management is used for storing generic user related profile information (e.g. age, impairments, preferences) as well as service specific user data.

This section contains the PM data model, design, and interface description. Requirements have been omitted, since the only consumer for the app show cases will be the personalization engine (PE) in WP5. The PE is developed together with the PM, and thus a common description will be part of the deliverable D5.3.2.

5.1. Data model

This section contains the data model.

The data model field obligation is indicated with 'M/O' - meaning 'mandatory/optional'. The amount of objects is indicated in the occurrences column with '1/n' - meaning 'unique/multiple'.

5.1.1. User Profile

The profile data model is shown in Table 21.

Level 1	Level 2	Level 3	JSON Type	JSON Object (preliminary)	Obligation	Ocurrences
Node			object		M	1
Identifier (unique)			object		M	1
	ID		string	id	M	1
	Alias		string	alias	O	n
	UUID		string	uuid	M	1
Meta			object	meta	O	1
	Description		string	description	O	1
	Success indicator		boolean	success	O	1
	User who created the entry		string	created_by	O	1
	Date of creation		datetime	created_at	O	1
	Date of last update		datetime	updated_at	O	1
	API version		string	version	O	1
Parameters						
	Attributes		object	attributes	O	1
		Attr. n	object		O	n
Services			*object*		*O*	*n*
	Service ID		*string*	*service*	*M*	*1*
	User ID in service		*string*	*suid*	*O*	*1*
	Permissions		*object*	*permissions*	*O*	*1*
	Attributes		*object*	*attributes*	*O*	*1*
		Attr. n	*object*		*O*	*n*
Links			object	links	M	1
	Users		array	users	M	1
		User ID	number	id	M	1
		API link	string	href	M	1

API version:	3
Date:	17.5.13

Table 21 User profile data model

The data model may be extended for future API versions.

The service related profile information will not be part of the implementation (neither web front-end nor API).

For providing the highest flexibility, service parameters in the profile and service definitions are decoupled. Thus, the service ID is used to link to the service profile. Profile dependent parameters can be added if necessary

5.1.2. Service Profile

The service data model is shown in Table 22.

Level 1	Level 2	Level 3	JSON Type	JSON Object	Obligation	Ocurrences
				(preliminary)		
Identifier (unique)			object		M	1
	ID		string	id	M	1
	Alias		array	alias	O	n
Services			object		M	1
	Name		string	name	M	1
	Address		string	address	M	1
API version:	1					
Date:	08.09.2012					

Table 22. Service data model

The data model may be extended for future API versions.

The service profile defines services that the user profile references to. For now, it specifies the address of the service. Parameters can be added if necessary.

5.2. Design

The module consists of front-end, back-end, and database layers.

The front-end is an HTTP web service in the form of a RESTful API. The back-end contains the module logic and communicates with the data base layers.

5.2.1. Database implementation

5.2.1.1. Description

The data model shows the two main domains:

- Profile
- Service

Figure 16 in section **Chyba! Nenašiel sa žiaden zdroj odkazov.** shows the relations between the domains in both identity and profile management.

The profile is referenced in the user data.

5.3. Interface

5.3.1. Internal Implementation

The profile management module has been implemented acc. the design that is described in section 5.2.

5.3.2. API

The basic functions of the profile management component are described in D6.1.1, sec. 6.3.5 [41]. The mentioned functions in this deliverable are not elaborated and mapped to the API.

In case no parameters are added, this shall be indicated by inserting 'n/a' (not applicable) into the table.

Data types for the parameters are stated in section 5.1 covering the data model.

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete profile in the system**		
Function:	Retrieve profile		
Method:	GET	**Resource:**	/profiles/{profileid}
Parameters:			
profile id	The id of the profile that shall be retrieved (linked in user struct).		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete profile in the system**		
Function:	Add profile		
Method:	POST	**Resource:**	/profiles
Parameters:			
BODY	The body of the message may contain profile parameters acc. [sec. 5.1.1]. These parameters are provisioned to the created user profile.		

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete profile in the system**		
Function:	Modify profile		
Method:	PUT	**Resource:**	/profiles/{profileid}
Parameters:			
profileid	The id of the profile that shall be modified.		

BODY	The body of the message shall contain profile parameters acc. [ref. this doc, sec.]. These parameters are provisioned to the profile that shall be updated. They will override existing parameters.

Function (Ref. D6.1.1):	**Retrieve, add, modify, delete profile in the system**		
Function:	Delete profile		
Method:	DELETE	**Resource:**	/profiles/{profileid}
Parameters:			
profile	The id of the profile that shall be deleted.		

The resource that is mentioned in the API description is adhered to the root URL, for example:

- Root URL: https://pm.example.org/api/v1/
- Full URL for first case: https://pm.example.org/api/v1/profiles/a1b2c3

The response code will provide general information about the success of the request. The response headers carry information relevant to the request and further requests. The response bodies of all requests will contain the complete data sets.

Data can be searched using the search string for the value in the form /search/profiles/id=abc12345.

Unless specified otherwise, the default content is used in the body. The requested resource might specify the content type in case multiple content types are offered (e.g. /profiles.json or /profiles.xml).

5.4. PM Requirements

The profile management has no specific requirements. It will be modelled according the requirements of the personalization engine and described in deliverable D5.3.2.

The profile management will offer an option to retrieve the profile according to the user ID (i.e. search). The IdM may be extended to have this link present as well as part of the user profile.

Interactions between IdM, PM, and PE will be shown in deliverable D3.3.2.

6. Security Manager

This chapter describes high level design for implementation of Security Manager into HBB Next project. It is designed with following key aspects:

- Provide key&certificate management-related functions within HBB-NEXT domain
- Provide privacy and access control for users
- Be platform independent
- Support open APIs for smooth integration of current or future HBB-NEXT technology building blocks
- Support any of present and future multimedia HBB-NEXT services

Please note that this document describe design of proposed implementation that is specific for functional prototype. Any other design specifics that are beyond of the prototype focus is subject of another document. However specific prototype design described here is fully align with the general Security Manager design.

6.1. Scope

The Security Manager (SM) component is responsible to:

- manage multi-factor authentication, authorization, and policy enforcement for different levels of privacy and for profile data access control,
- handle authentication, i.e. it acts as an identity provider towards the STB. The IdM shall act as back-end for the authorization process,
- manage and verify tokens,
- provide PKI management for HBB Next domain: Certificate Authority (CA) and Certificate Revocation List (CRL),
- store of security related data (pass phrases, certificates, keys, tokens, …),
- grant secure access of the HBB Next local services to the external networks,
- update/retrieve certain data in IDM.

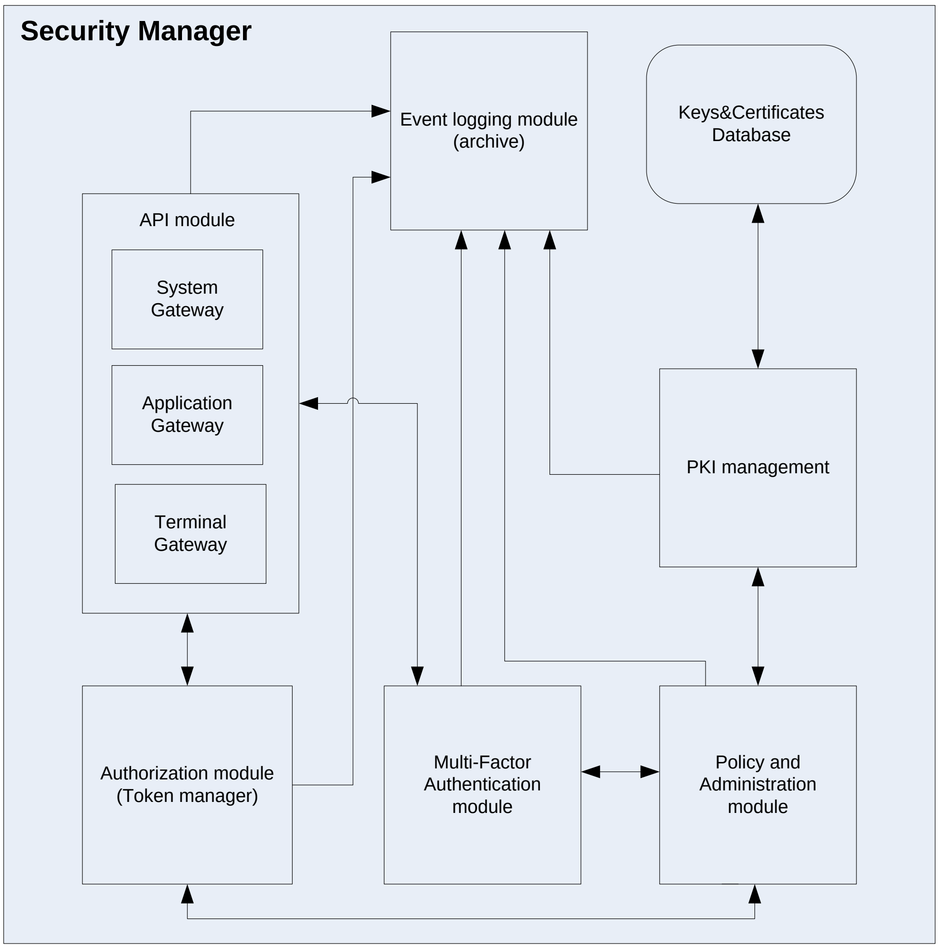

Security Manager
Event logging module (archive)
Keys&Certificates Database
API module
System Gateway
Application Gateway
Terminal Gateway
PKI management
Authorization module (Token manager)
Multi-Factor Authentication module
Policy and Administration module

Figure 17. Internal architecture of the Security Manager

The Security Manager consists of several modules:

Authorization module (Token manager) – will evaluate if the terminal has access to the HBB network as role-based access controller. This module will either grant or deny access to the HBB domain.

Multi-Factor Authentication (MFA) module – will be responsible for managing multi-factor authentication processes, depending on required level of security during User/Group is accessing the application/s.

MFA will perform a user identification process to create a list of "best matches" and then perform a series of verification processes to determine a conclusive match. Number of necessary verification processes depends on the required authentication level for accessing the service/application/data. HBB-NEXT project supposes to have several required levels of authentication to verify users before accessing the service with required “level of security”.

(Personal EPG – low level is required; instant messaging – middle level is required, e-banking – high level is required)

Policy and Administration module – policy enforcer for MFA module and point of security rules mapping and configuration.

PKI management – will be intended to manage Public Key Infrastructure (PKI) tasks within HBB-NEXT domain, acting as certificate management system within HBB-NEXT domain.

Event logging module (archive) – is event logger for all activities performed by SM.

Keys&Certificates Database – will hold sensitive cryptographic key information and single public key certificates.

API module – secure interface for communication among HBB-NEXT entities consisting of three different sub-modules, the so-called gateways.

- *System Gateway* – an interface towards HBB-NEXT core modules (Identity Management, Profile Manager, Trust & Reputation)
- *Application Gateway* – an interface towards HBB-NEXT application, which will be used only for communicating with Application Servers and not to applications hosted on the terminal. *Note – necessity of this interface will be a part of further detailed design analysis within WP3.*
- *Terminal Gateway* – an interface towards Terminal/ End-user device. This Gateway is needed because of different rules for external devices/terminals might be used than to all servers within Core and Application layers.

6.1.1. Solution Overview

6.1.1.1. Depended entities

Identity Management (IdM) - This module is responsible for Identity Management role in HBB Next environment. It stores and manages all user related identity data and provides it for other entities, excluding security related data (i.e. passphrases, secrets, certificates, keys …). Security Manager updates respective data in the IDM module to keep it actual.

Multimodal Interface (MM) - This module is responsible for identification and recognition of subjects that are presented in certain area either by its voice or by its visage. Additionally, for security purpose, MM module provides to the Security Manager data that are subject of additional check to confirm (or increase) security level of the user. Security Manager might enforce various ways how to prove user's identity, based on the situation or following action needs.

EPG - This module offer Electronic Program Guide (EPG) for the HBB Next service users. Based on the metadata and related services, it generate data that are sent to the display unit. One aspect that influence process of asset generation is the content that actual user is able to view or follow. Here the Security Manager fulfils the role of supplying engine providing to EPG Service information whether this user is authenticated enough to be authorized to view certain content. Authentication level is determined with cooperation of MM module.

Facebook FrontEnd application (FB FE app) – This module is described for prototype purpose only. FB FE app represents here general social network service that is used by HBB-NEXT users. The Security Manager is responsible for the secure access of FB FE app to the FB app central service, by granting access via distribution of tokens. Security Manager also provides service whether certain user is obliged to use such service (authenticated enough) and manage triggering of additional multifactor authentication.

The above mentioned modules are planned to be used for demo purposes, but all other application modules/entities will depend on Security Module policy enforcement, thus shall be mentioned here.

6.1.2. Management

The Security Manager might be connected to the management system of respective provider. This system is beyond of the scope of this design document and is subject of specific provider related integration. The general requirement is to connect the provider's own higher level OSS/BSS via a firewall that also does NAPT.

6.1.3. Hardware architecture

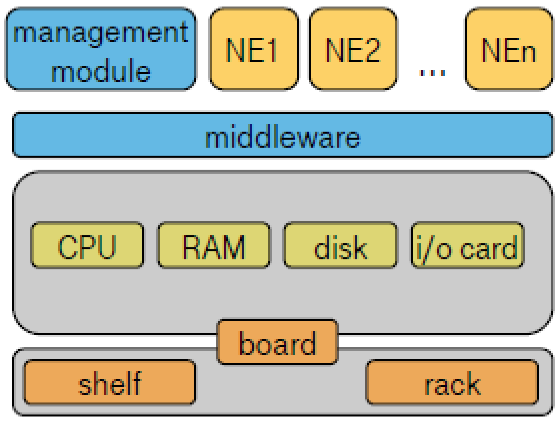


Figure 18. Security Manager - Overview of hardware architecture

Security manager is designed as HW platform independent component that must follow criteria:

- CPU: 2.2GHz dualCore
- RAM: 2GB
- HDD: 250GB
- Network: 100 Mbps ETH

6.1.4. Authorization model

Authorization is important part of overall security in the HBB Next domain, especially for all terminals, such us set-top-boxes or mobile HBB next-enabled devices as well as all modules which requires secure APIs communication.

Authorization module as a part of Security Manager takes the responsibility to grant or deny access to the HBB next domain for all terminals and together with authentication shall take the role of Provider for Centric Identity. Provider of Centric Identity is not to be used as Identity Management for the users of HBB next services, but for the HBB Next terminals and all modules which are in direct communication with Security Module (through the secured APIs) or requires secure APIs in between.

Model will be based on the OpenID and OAuth standards together with the Public Key Infrastructure (PKI) architecture.

Note: Authorization model is part of further design analysis, where all details will be described.

6.1.5. PKI architecture

This chapter describes the Public Key Infrastructure (PKI) design within the HBB Next domain. The components of Security Manager module responsible for certificate generation, revocation, lifecycle and key distribution management are highlighted in Figure 19.

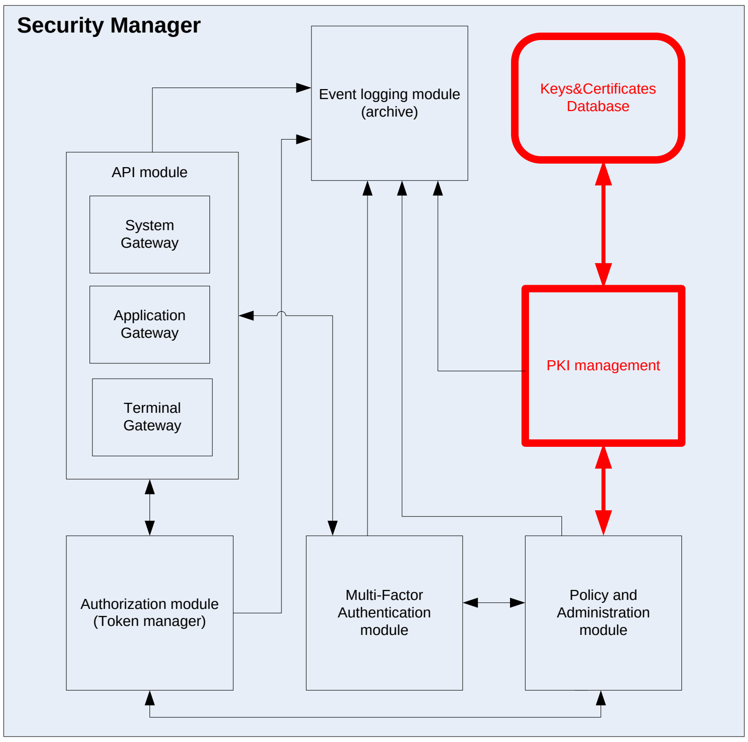

Security Manager
Event logging module (archive)
Keys&Certificates Database
API module
System Gateway
Application Gateway
Terminal Gateway
PKI management
Authorization module (Token manager)
Multi-Factor Authentication module
Policy and Administration module

Figure 19. Overview of responsible modules

PKI Management and Keys & Certificates Database plays role of Public Key Infrastructure internally, for one HBB Next domain. This chapter describes internal structure of those components as well as the whole enrolment and revocation strategy, including the details about key distribution within the "home" HBB Next domain. Relations between HBB Next domains are not part of this chapter and will be analysed separately.

PKI management is a place of certificate authorities – Certificate Authority and Registration Authority. Optionally, it can host a responder for certificate status, if Certification Revocation List (CRL) is not used.

Usage of CRL or Online Certificate Status Protocol (OCSP) or combination of themis subject of further design analysis. This deliverable provides first designs and protocols for the following components:

6.1.5.1. PKI Management

PKI Management is one of the modules of the Security Manager, used to create, distribute, store and revoke digital certificates within the HBB Next domain.

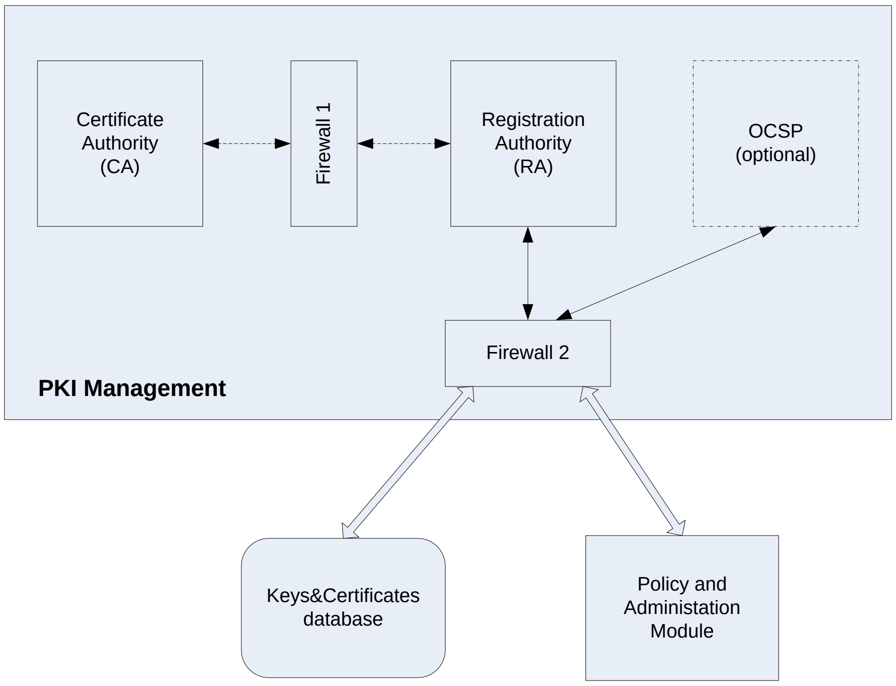

Certificate Authority (CA)
Firewall 1
Registration Authority (RA)
OCSP (optional)
Firewall 2
PKI Management
Keys&Certificates database
Policy and Administation Module

Figure 20. PKI Management – internal architecture

Certificate Authority (CA)

CA is a trusted certificate issuer and the top of trusted hierarchy within the HBB Next domain. CA will be strictly separated and wouldn't be networked with any other component, except with the associated RA. Connection will be separated by an internal firewall, protecting the CA.

Registration Authority (RA)

The RA is used for modest security level, which means that only the RA can forward certification request to CA within the HBB Next domain. Otherwise, isolated CA approach would be used (highest security level), and human intervention is needed.

The RA will only communicate with the certificate requester and post a copy of a certificate to an LDAP directory, located in the Keys&Certificates database.

Online Certificate Status Protocol (OCSP) sever

The OCSP server is used for checking the revocation status of the already issued certificates. This component does not have to be presented if Certificate Revocation List stored in LDAP is used. Both methods, OCSP vs CLR, has pros/cons and must be evaluated in further design.

Usage of OCSP is marked as optional for now.

Firewall 1

A logical component which strictly defines Access Control (ACL) rules, protects the CA and only allows communication between the CA and the associated RA.

Firewall 2

A logical component protects whole PKI Management module used for communication towards other related components of the SM.

PKI management shall be located on physically separated server.

6.1.5.2. Keys&Certificates Database

Keys&Certificates database represents central storage for keys, certificates and CRLs used for PKI Management, based on LDAP service.

The component is protected with firewall and shall be connected to PKI module only.

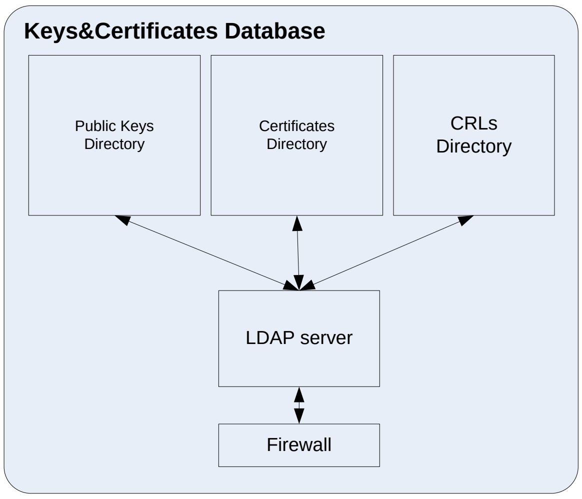


Figure 21. Keys&Certificates database

6.1.6. Use case

Below mentioned use case shows role of SM as well as other components within HBB-NEXT architecture.

6.1.6.1. Preconditions:

- Lisa - owns HBB-NEXT enabled mobile device or tablet,
- Tom - is Lisa's brother. He has different interests than his sister and has no mobile device,
- They have at home HBB-NEXT enabled device – TV&STB (it may be a single device). This device is connected to the Internet,
- Lisa's and Tom's parents – they are setting rules on TV&STB HBB-NEXT device for their kids
- All devices are connected to the Internet (for the purposes of this scenario),
- Mia – Lisa's friend, she is using other HBB-NEXT provider (other HBB-NEXT domain)
- Profiles of Lisa and Tom will be pre-filled (for demo only). Both are using one HBB-NEXT provider
- Lisa is using HBB-NEXT service and her tablet is connected to internet
- Her profile shows that she has brother Tom and also Tom's photo (demo case – filled before)
- Lisa will use applications that demonstrates the applicability of multifactor-authentication, concretely:
    - *EPG* – Lisa's customized program guide – Level 1 - sufficient identification is face / voice recognition
    - *Facebook – Level 2* – say a passphrase or password if nobody else is in the room (this will be always monitored) OR PIN/Passcode with using of remote control if someone else is in the room (saying of passphrase will be disabled)
    - *Messaging – Level 2* - same conditions as for Facebook app

  - *<u>Shopping – before payment</u>* - say a passphrase or PIN. The system shall also look for someone else in the room (this could be detected by MM (multi-modal interface) and IDM service has announced Shopping)
  - *<u>Shopping- payment phase – Level 3</u>* - Lisa wants to pay selected items in the shop - SMS confirmation code or pupil recognition is required (it must be discussed with Gregor)

- same applications will be available through TV&STB

6.1.6.2. Scenario:

1. Lisa is in the living room and has its own programs and videos available through the VoD and EPG menu and she is chatting with Mia (chat service is triggered via MM with the profile of Lisa).

2. Tom is coming and is identified by camera on TV&STB. Screen shows – "Tom is coming" Now, EPG expands by their common preferences (as an example). Running IM conversation between Lisa and Mia is moved to the background and is now available on the Lisa's tablet only. If Lisa wants to continue with the chatting, she will be asked to enter a PIN via the remote control.

3. Tom on launching Facebook app on the TV&STB - because Lisa was there before, the system knows that no notifications are necessary, and so allows him to sign in (via PIN/passcode) even presence of Lisa.

4. Lisa and Tom are in the living room debate on video that they saw. System offers them the opportunity to see this video with similar content because it was discerned from conversation (Given that this option is enabled Tom and Lisa's profile, and be able to ban permanently)

5. Tom in on the Facebook / (or other app) and wants to buy a tree for his virtual farm. It is forbidden to him because lack of his age. Purchase can be enabled by parents only, thus they are receiving alert message (only parents and persons authorized by them shall be able to allow their children to buy something.

End of demo.

6.1.7. Sequence Diagram

This chapter describes communication of all components involved in the prototype use case, described in the chapter above. We use sequence diagram as self-explanatory form of description.

Main simplification used here:

- We suppose that Security Manager is already pre-provisioned with respective data, i.e. IDs, pass phrase, trust level data, …

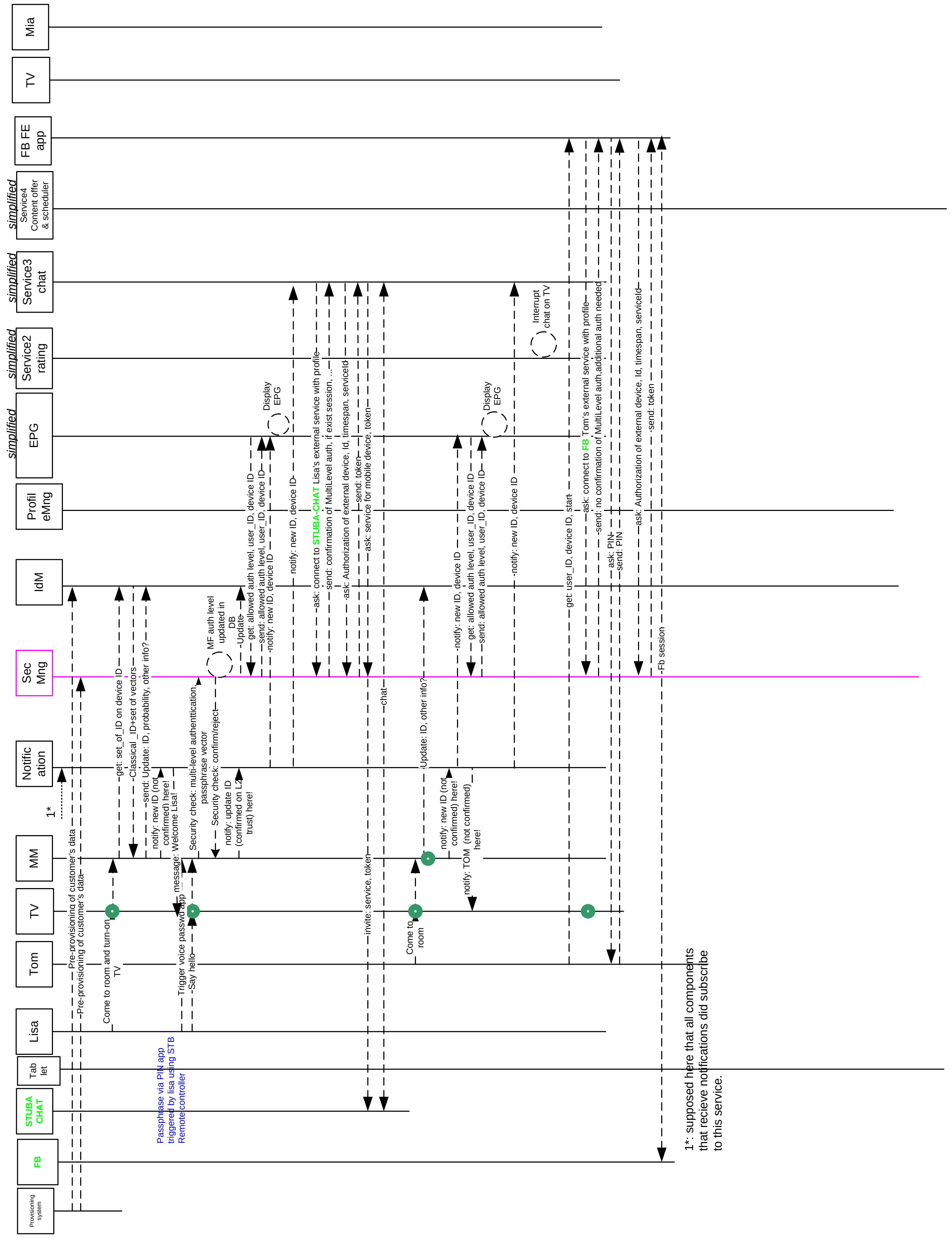

Mia
TV
FB FE app
simplified Service4 Content offer & scheduler
simplified Service3 chat
simplified Service2 rating
simplified EPG
Profil eMng
IdM
Sec Mng
Notific ation
MM
TV
Tom
Lisa
Tab let
STUBA CHAT
FB
Provisioning system
Pre-provisioning of customer's data
Pre-provisioning of customer's data
1*
Come to room and turn-on TV
get: set_of_ID on device ID
Classical _ID+set of vectors
Update: ID, probability, other info?
send: Update: ID, probability, other info?
notify: new ID (not confirmed) here!
message: Welcome Lisa!
Trigger voice passwd app
Say hello
Security check: multi-level authentication passphrase vector
Security check: confirm/reject
notify: update ID (confirmed on L2 trust) here!
MF auth level updated in DB
Update
get: allowed auth level, user_ID, device ID
send: allowed auth level, user_ID, device ID
notify: new ID, device ID
Display EPG
notify: new ID, device ID
ask: connect to STUBA-CHAT Lisa's external service with profile
send: confirmation of MultiLevel auth, if exist session, ...
ask: Authorization of external device, Id, timespan, serviceId
send: token
ask: service for mobile device, token
invite: service, token
chat
Come to room
Update: ID, other info?
notify: new ID (not confirmed) here!
notify: TOM (not confirmed) here!
notify: new ID, device ID
get: allowed auth level, user_ID, device ID
send: allowed auth level, user_ID, device ID
Display EPG
notify: new ID, device ID
Interrupt chat on TV
get: user_ID, device ID, start
ask: connect to FB Tom's external service with profile
send: no confirmation of MultiLevel auth,additional auth needed
ask: PIN
send: PIN
ask: Authorization of external device, Id, timespan, serviceId
send: token
Fb session
Passphrase via PIN app triggered by lisa using STB Remotecontroller
1*: supposed here that all components that recieve notifications did subscribe to this service.

Figure 22. Sequence Diagram

6.1.8. API description

Security Manager API component communication is based on Restful API,based on HTTP GET requests and JSON respond flow.

HBB-NEXT API is divided into the following modules: Core systems, Terminals, Applications and Management. This document contains a section for each one.

6.1.8.1. API for terminals/services

This section describes the interactions between the security manager and HBB-NEXT terminals (user devices) or services (parts of the platform, such as EPG).

Albeit the aforementioned are two different modules, they are functionally very similar, thus they will both be described in this section. Security manager handles all of these requests and this protocol serves as the main function of the manager.

Overview

HTTP POST Requests are sent from end terminals (TV sets, tablets, etc.) or from other services using the HBB-NEXT platform. These requests have a mandatory parameter “ID”, which identifies the request for SecM. The rest of the parameters depend on the request itself. Malformed or unrecognized requests result in error responses.

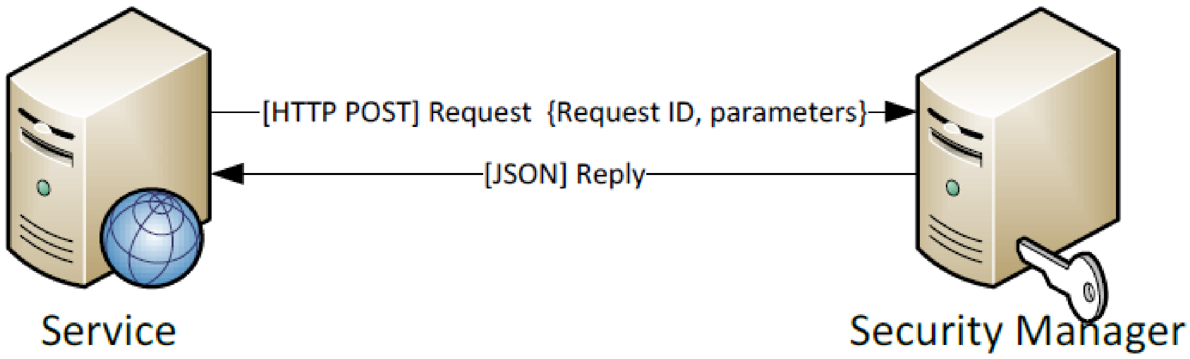


Figure 21: Communication flow between the security manager and a service

Protocol

The protocol is divided in requests and responses. The requests are sent as HTTP POST messages, replies are formatted as JSON strings sent over HTTP.

Requests

1. Login user

Request originates from user terminal wishing to log in and receive a security token. User ID and Device ID are mandatory parameters. In case no PIN is entered, the field must be filled with "null". Login request takes in pin/password inside the request and MMI data from IdM.

id = 1 , user ID (int), pin (number), device ID (int)

Medium	Parameter	Content
Body	id	1
Body	user	User ID value
Body	pin	User PIN or "null"
Body	device	Device ID

Example: id=1&user=1&pin=null&device=2

2. Authorize token for certain level

SecM looks up current security level, input from MMI and decides on next steps. It either grants authorization, or requests further action.

id = 2 , token (string), level (int), apptoken (string)

Medium	Parameter	Content
Body	id	1
Body	token	Token string
Body	level	Queried level (integer)
Body	apptoken	Application token

3. Application token setup

This requests is for applications, which require an initial token. id = 3, applicationID (int)

Medium	Parameter	Content
Body	id	1
Body	application	Application ID value

Responses

1. User logged in successfully

SecM generates a token and sends it to the entity which requested it. SecM appends a signed token

```
{“ID”:1,“user”:string,“device”:string,”token”:string,“level”:int}
```

2. Token authorized for level X

SecM authorized the token for the service at certain level

```
{“ID”:2,“level”:int}
```

3. Application token set up

Token was granted for an application

```
{“ID”:3,“token”:string}
```

4. Invalid credentials

```
{“ID”:11,“user”: string}
```

No information if username or password/pin was incorrect as a security measure

5. Unable to process request

This is a generic error for ALL undefined cases.

```
{“ID”:12,“errno”:int}
```

6. Insufficient privilege level (minimum required is X)

Level is set to the current token level, if nonexistent then set to 0.

```
{“ID”:13,“token”:string,”level”:int}
```

6.1.8.2. API for Applications

Overview

The application API is handling interactions between the security manager and various applications (integrated or 3rd party), that access data in IdM. SecM

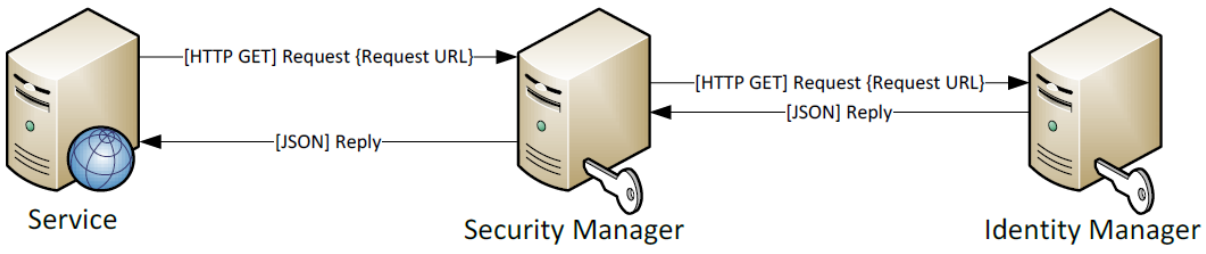

[HTTP GET] Request {Request URL}
[JSON] Reply
[HTTP GET] Request {Request URL}
[JSON] Reply
Service
Security Manager
Identity Manager

Figure 22: Communication flow in data requests

Protocol

The protocol is derived from the IdM API. SecM recognizes three different requests: Request for user data, request for device data and request for contextual data (data such as currently active users).

Application requesting data is submitting HTTP GET requests in a format similar to one used for Identity Manager. Applications are viewed as clients in this API dialogue. The application protocol follows a REST structure, so all requests are in a unified format of *http:/IP:8080/class/id/?access_token=token .*

The aforementioned *access_token* parameter is a user security token. IP is the Security Manager IP address (or URL), 8080 is the access port for requests.

This security procedure is not working in version 0.2.

SecM then responds with a string obtained from IdM, with certain parts censored out, depending on the security level of the submitted token. If current security level of the token is insufficient, some fields in the JSON string will be replaced with message “SecurityLevelX”, where X is minimal security level for the field to be revealed.

Some fields such as user PIN, password, and MMI values will never be revealed, and are marked as “SecurityLevel9001”.

Users

Client submits a request to the users branch, followed by an ID of certain user.

Example: *http:/127.0.0.1:8080/users/2/?access_token=token*

Devices

Client submits a request to the devices branch, followed by an ID of certain device.

Example: *http:/127.0.0.1:8080/devices/3/?access_token=token*

Contexts

Client submits a request to the contexts branch, followed by an ID of certain context.

Example: *http:/127.0.0.1:8080/contexts/1/?access_token=token*

6.1.8.3. API for core systems

These interfaces are not defined by the Security Manager. Communication protocols used are defined by actual systems SecM is communicating with.

API for IdM

A basic REST structure for item access. Functions include accessing certain users or devices, changing their parameters or creating new ones (registration). Actions are sent via HTTP requests; replies are formatted as HTML or JSON.

API documentation is available separately, but for purposes of this document a brief introduction is included.

Communication with IdM is handled with HTTP GET requests on the IdM API, which responds with JSON strings.

The IdM supplies three basic objects, the User object, the Device object and the Context object. Each is called by accessing the REST structure of the API. The base root of the API is set to the address “http://hbbnext.ngidm.org/api/v2/”. Each type can be accessed in its own branch in the REST structure. Each object is invoked by the parameter Device ID.

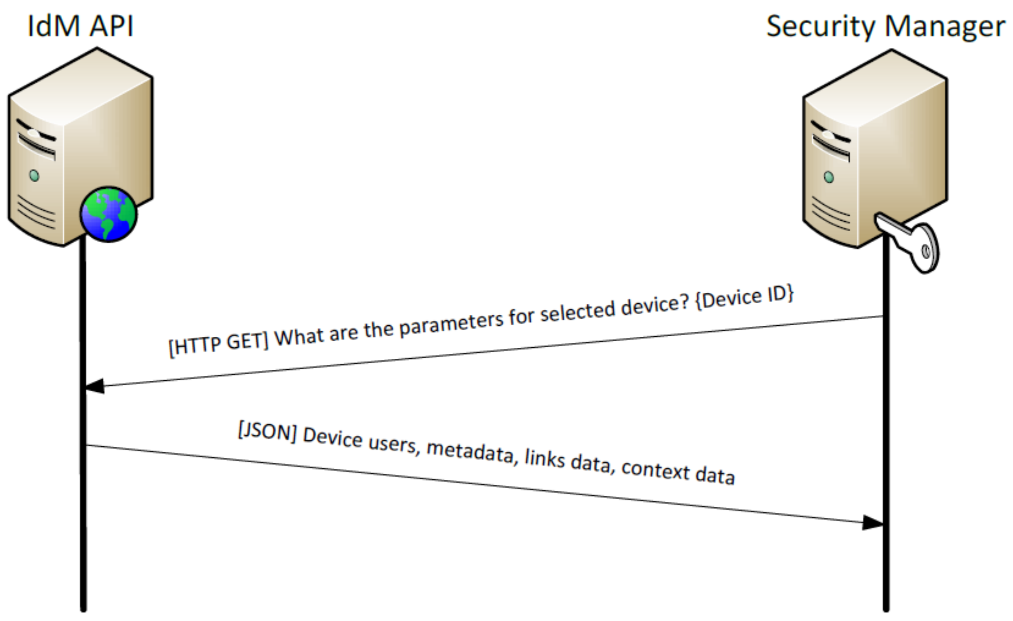

IdM API
Security Manager
[HTTP GET] What are the parameters for selected device? {Device ID}
[JSON] Device users, metadata, links data, context data

Figure 23: Security Manager requesting device parameters

Devices

After placing a request on the /devices/ branch, we can request an object containing data for the selected device. The devices object is required for the security manager for the list of all users currently active on the device. This object has the biggest impact for deciding later action in the security manager.

Users

The user objects can be found in the /users/ branch. This object contains the levels of authenticity of the user on which further security decisions are made.

Contexts

The context object can be found in the /contexts/ branch. It defines cross connections between active devices and users using the devices.

Management API

The setup for the security manager contains three entities – a management station, web server and the security manager application. The web server is hosting a REST API from which the security manager receives data.

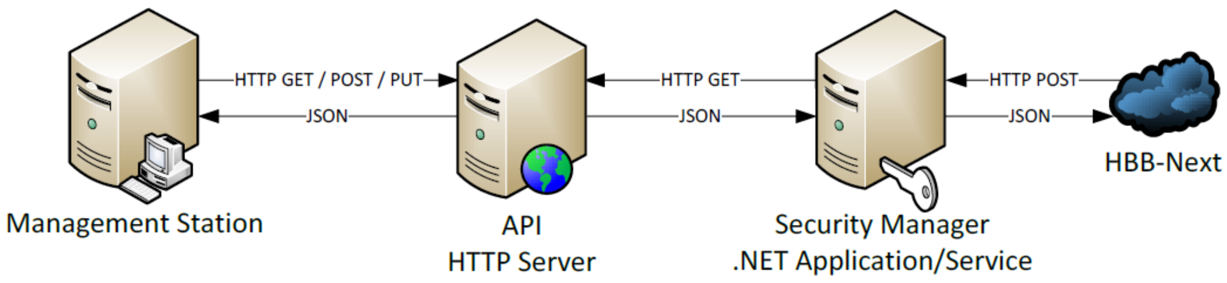


Figure 24: Communication entities and flows

Overview

Management API for security manager is a REST structure. Its function is tied to the management module of SecM. In the current version, the options are to edit multimodal and timer values. The prototype supports modifying three available security levels, and is listening at port 3000.

Actions

API uses HTTP requests, responds with JSON or Cookie. Reading is done via HTTP GET, editing is conducted via HTTP POST. This string contains parameters for the queried policy level packed into an object. It's also possible to query for a single value as well. For maintenance access, data can be viewed as an HTML page.

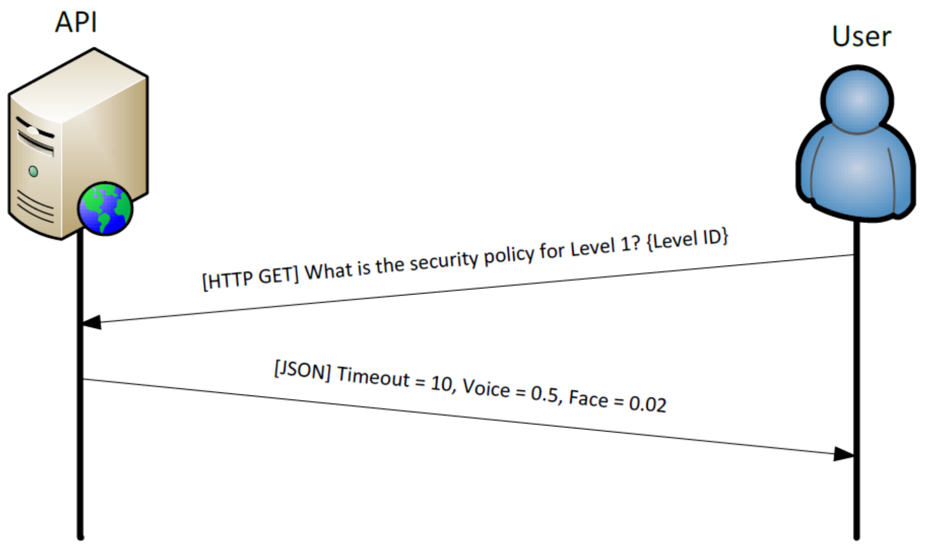

API
User
[HTTP GET] What is the security policy for Level 1? {Level ID}
[JSON] Timeout = 10, Voice = 0.5, Face = 0.02

Figure 25: Using HTTP GET to obtain a resource from a REST structure

Values are edited by sending a HTTP POST request to /levels/x with these parameters: (x being a number)

Medium	Parameter	Content
Cookie	_mgmt1_session	BAh7B0kiD3Nlc3Npb2fa684 (varies)
Body	utf8	%E2%9C%93
Body	_method	Put
Body	authenticity_token	MNQe6yf8XEK6Q/19mHVP28= (received from <meta content> tag)
Body	level[timer]	Value in msec
Body	level[voice]	Value (0.00 to 1.00)
Body	level[face]	Value (0.00 to 1.00)
Body	commit	Update+Level

Table 23: Available parameters

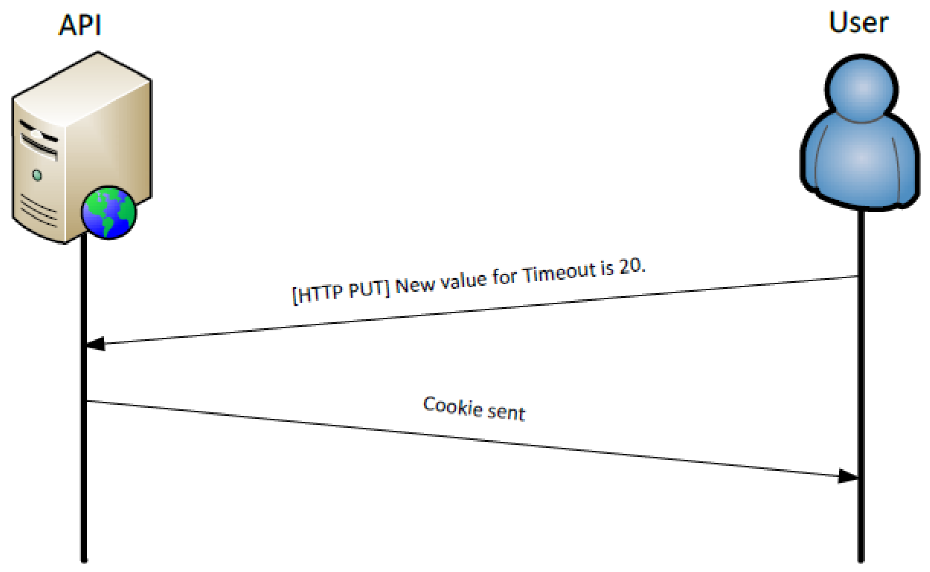


Figure 26: Updating an entry via HTTP PUT

The option of adding new / deleting existing has been removed from the prototype.

6.1.9. Requirements on HBB-NEXT architecture

1. Set-Top Box must have an API call to pop up “enter PIN” screen for the user. This screen must be tied (forward data) to Security Manager, and must display which application/service requested the PIN. PIN response should also be signed with device private key.

Reason: if applications handle PIN transfers to security manager, confidentiality of the user is violated.

2. After PIN is entered, last action from the user should be repeated (such as submitting a form in appstore again). If the decision is that this should be transparent, then the set-top

box should repeat the last action before PIN was submitted. If user declines to submit his PIN, then the last action should not be repeated.

Reason: a new token is generated for the user after (s)he enters PIN code. the application needs to know the new token.

3. If 3rd party applications are allowed to use centralised payment mechanism (such as purchasing goods / apps) via the HBB-NEXT payment system, a central trusted payment gateway must be established, that would check whether the user really has the token level service requesting the transaction is stating.

If this module is created, then it must come with an API hook at the set-top box to process payments, in a way similar to credit card processing at a CC gateway and not directly at the shop's page.

Reason: if payments are processed at the 3rd party service, the service could state that the transaction was **NOT** successful even if it was, and request additional transactions.

6.1.10. Security levels

Basic concept of SecM in the demonstration phase supports 3 multi-factor levels with the following common assumptions:

- Concept of SecM always reacts "on request", which means that SecM is doing the decision making procedure only if the Application is requesting it.
- User must be registered before to use identification based on Level 1 and Level 2 . If not user, is automaticky asked for login&PIN (Level 3) procedure with direct forwarding to registration.
- If two or more users are identified at the same time then the SecM allows an access to the Application for only one User, always with the highest probability(voice).
- Group mode is not supported in demo version, thus only one user has granted access to the Application when more users are using HBB-NEXT enabled device at the same time.
- Identification is based on voice recognition and face recognition only

- Passphrase (as voice or gesture) and biometrics are not supported in demo version
- Percentages for probability(voice) and probability(face) can be changed by administrator of SecM
- SecM can skip over levels, based on defined policies

Level 1- IdentifyParam: Voice; probability(voice)

This level means that the User is identified by voice if the probability(voice) reached the defined modality. Decision-making procedure is done when the User is accessing the Application where the Level 1 is sufficient for the access.

Level 2 - IdentifyParam: Voice, Face; probability(voice) & probability(face)

The level 2 is used when the User was not successfully recognized on the Level 1 or the modality(voice) did not reach defined modality. The User is identified by voice and face if the combination of the probability(voice) & probability(face) reached the defined modalities. Both of these factors (voice and face) are necessary to reach this level, so the condition requires “AND” logic rule.

Level 3 - AuthParam:login&PIN

The level 3 is used when the User was not successfully recognized on the Level 1 “OR” Level 2 but is mainly used for accessing the Application or procedure, where the authentication with credentials is required. The Level 3 is based on login@PIN authentication method. “Login” might be required if:

- Identification of user on the Level 1 or Level 2 was not successful
- Identification of user was not required before (e.g. access to Application without previous knowledge of who is accessing it.)
- The User is unknown person

6.1.11. General Sequence Diagrams

The following picture shows general authentication procedure of user for a certain level. First part of the diagram explains the way how the device is accessing an HBB next domain. In the second part is shown how the user is authenticated when accessing the Application (Service). Sequence flow in this part explains what has to be done if user has reached already the required level or not.

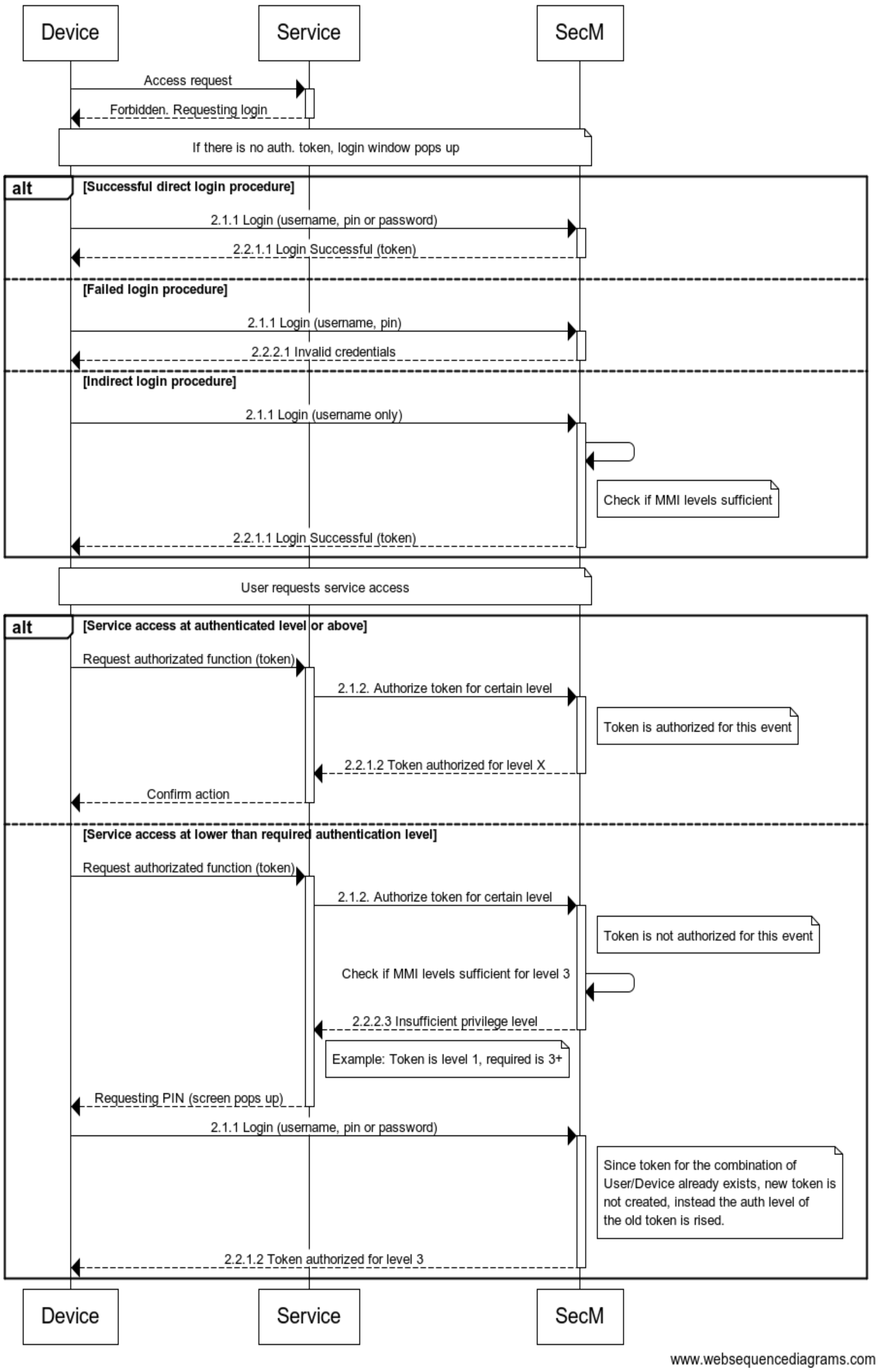

Device
Service
SecM
Access request
Forbidden. Requesting login
If there is no auth. token, login window pops up
alt
[Successful direct login procedure]
2.1.1 Login (username, pin or password)
2.2.1.1 Login Successful (token)
[Failed login procedure]
2.1.1 Login (username, pin)
2.2.2.1 Invalid credentials
[Indirect login procedure]
2.1.1 Login (username only)
Check if MMI levels sufficient
2.2.1.1 Login Successful (token)
User requests service access
alt
[Service access at authenticated level or above]
Request authorizated function (token)
2.1.2. Authorize token for certain level
Token is authorized for this event
2.2.1.2 Token authorized for level X
Confirm action
[Service access at lower than required authentication level]
Request authorizated function (token)
2.1.2. Authorize token for certain level
Token is not authorized for this event
Check if MMI levels sufficient for level 3
2.2.2.3 Insufficient privilege level
Example: Token is level 1, required is 3+
Requesting PIN (screen pops up)
2.1.1 Login (username, pin or password)
Since token for the combination of User/Device already exists, new token is not created, instead the auth level of the old token is rised.
2.2.1.2 Token authorized for level 3
Device
Service
SecM
www.websequencediagrams.com

Figure 23 General sequence diagram without an OpenID

The following figure explains in a similar way the situation (as displayed on the Fig.27) when the OpenID is used for accessing the HHB domain and accessing the Application (Service).

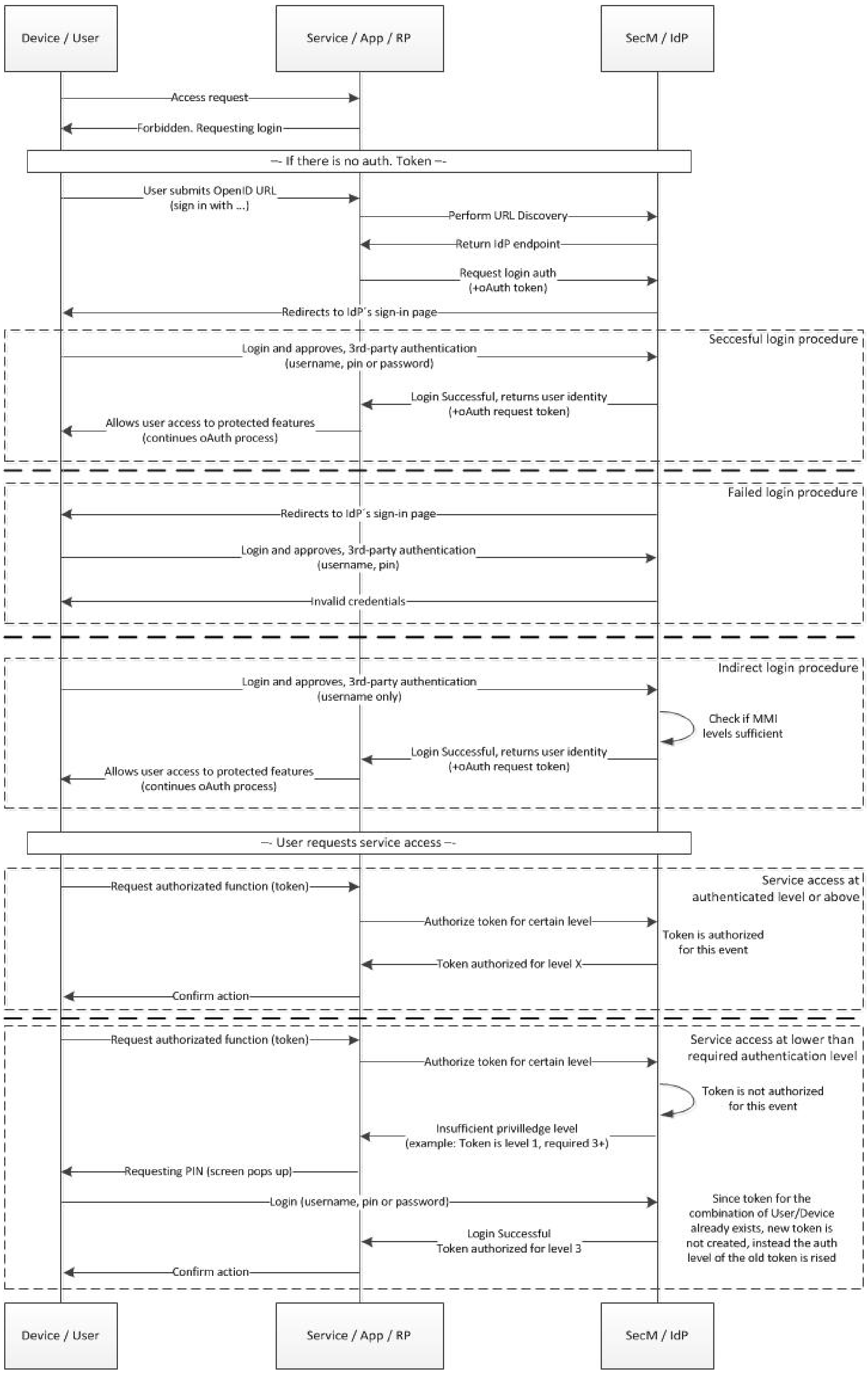

Device / User
Service / App / RP
SecM / IdP
Access request
Forbidden. Requesting login
-- If there is no auth. Token --
User submits OpenID URL (sign in with ...)
Perform URL Discovery
Return IdP endpoint
Request login auth (+oAuth token)
Redirects to IdP´s sign-in page
Seccesful login procedure
Login and approves, 3rd-party authentication (username, pin or password)
Login Successful, returns user identity (+oAuth request token)
Allows user access to protected features (continues oAuth process)
Failed login procedure
Redirects to IdP´s sign-in page
Login and approves, 3rd-party authentication (username, pin)
Invalid credentials
Indirect login procedure
Login and approves, 3rd-party authentication (username only)
Check if MMI levels sufficient
Login Successful, returns user identity (+oAuth request token)
Allows user access to protected features (continues oAuth process)
-- User requests service access --
Service access at authenticated level or above
Request authorizated function (token)
Authorize token for certain level
Token is authorized for this event
Token authorized for level X
Confirm action
Service access at lower than required authentication level
Request authorizated function (token)
Authorize token for certain level
Token is not authorized for this event
Insufficient privilledge level (example: Token is level 1, required 3+)
Requesting PIN (screen pops up)
Login (username, pin or password)
Since token for the combination of User/Device already exists, new token is not created, instead the auth level of the old token is rised
Login Successful Token authorized for level 3
Confirm action
Device / User
Service / App / RP
SecM / IdP

Figure 24 General sequence diagram with an OpenID

Security Manager creates a direct relation between the information being protected and the complexity of the authentication process.

From the user perspective the SecM combines factors that users already possess to use them for authentication purposes. Thus, the users are not limiting to password-only mechanism when accessing their favourite services within HBB Next domain. Introduced Multi-factor authentication with several levels of security could create highly-secure methods while the user is not irritating with complicated authentication steps.

SecM represents a new central point of security rules enforcement within HBB Next domain.

7. Conclusion

In this document the first outputs of WP3 have been described: Applications like voice identifications, 2D face recognition and Reputation framework have been demonstrated within first beta versions. Others, like Identity manager, Security Manager and Profile Manager have been described in details from system and design point of view. In the next year of project all applications will be enhanced and step by step integrated into one demo application. Faster classification methods will be implemented and tested so the recognition process would be able to run for an infinite time interval. New decision taking algorithms will be suggested to meet multi speaker particularities with a possibility of detecting unknown speaker (not being recorded in the database) by employing some confidence criteria or by creating general speaker model. There is an on-going research yielding to implementation of a gesture recognition system based on a pre-defined database. We plan to build a more sophisticated system which will recognize not only static gestures but also dynamic gestures and their combination. The preliminary documentation to the present applications has been described in this document.

8. References

[1] D2.2 System-, Service-, and User Requirements", http://www.hbb-next.eu/index.php/documents

[2] D. A. Reynold, "An Overwiev of Automatic Speaker Recognition Technology", in Proc. Of International Conference on Acoustic, Speech, and Signal Processing, Orlando USA, 2002

[3] F. Bimbot, J-F Bonastre, C. Fredouille, G. Gravier, I. Magrin- Changolleau, S. Meignier, T. Merlin, J. Ortega-Garcia, D. Petrovska-Delacretaz, D. A. Reynolds“, A tutorial on text independent speaker verification", Journal on Applied Signal Processing, Hindawi, pp. 430-451, 2004

[4] T. Kinnunen, H. Li, "An overview of text-independent speaker recognition: From features to supervectors", Speech Communication vol 52, 2010, pp.12–40, ISSN 0167-6393

[5] F. Hönig, G. Stemmer, Ch. Hacker, and F. Brugnara, "Revising Perceptual linear Prediction (PLP)", Proceedings of INTERSPEECH 2005, pp. 2997-3000, Lisbon, Portugal, Sept., 2005

[6] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech", J. Acoust. Soc. Am, Vol. 87, No. 4, pp. 1738-1752, 1990

[7] T. M. Mitchell, Machine Learning, McGraw-Hill, ISBN 0-07-042807-7, 1997

[8] A..A. Miranda, Y.A. Le Borgne,G. Bontempi, New Routes from Minimal Approximation Error to Principal Components, Neural Processing Letters, Vol. 27, N. 3, Springer, June 2008

[9] H. Gao, J. W. Davis, Why direct LDA is not equivalent to LDA, Patern recognition, vol. 39, pp. 1002-1006, Nov. 2005

[10] Oravec Miloš, Mazanec Ján, Pavlovičová Jarmila, Pavel Eiben,FedorLehocki: Face Recognition in Ideal andNoisyConditionsUsing Support Vector Machines, PCA and LDA, Face Recognition (Ed. Milos Oravec), ISBN: 978-953-307-060-5, IN-TECH, 2010

[11] X.Tan, S.Chen, Z.-H. Zhou, and F. Zhang, Face Recognition from a Single Image per Person: A Survey, Pattern Recognition, 39(9), pp.1725-1745. 2006.

[12] Oravec,M., Pavlovičová,J., Mazanec,J., Omelina,Ľ., Féder,M., Ban,J.: Efficiency of Recognition Methodsfor Single Sample per Person Based Face Recognition, chapter in monograph Reviews, Refinementsand New Ideas in Face Recognition (Ed. Peter M. Corcoran), ISBN 978-953-307-368-2, IN-TECH, Croatia, 2011, pp. 181-206

[13] R. Verschae, J. Ruiz-del-Solar, and M. Correa, "Face recognition in unconstrainedenvironments: a comparativestudy," in Proceedingsofthe Workshop on Faces in Real-Life Images: Detection, Alignment, and Recognition (ECCV '08), pp. 1–12, Marseille, France, October 2008

[14] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," Pattern Analysis and Machine Intelligence, 2011.

[15] OASIS Open Reputation Management System (ORMS) Draft Version 0.1, availableathttp://wiki.oasis-open.org/orms/WorkingDraft

[16] D5.2 DESIGN AND PROTOCOL: Multimodal Interface and Context Aware Recommendation Engine; http://www.hbb-next.eu/index.php/documents

[17] D2.4 Description of the Selected Business Model, http://www.hbb-next.eu/index.php/documents

[18] D3.1 ANALYSIS: State of The Art on Identity, Security and Trust, http://www.hbb-next.eu/index.php/documents

[19] D6.1.1 Initial Version of the HBB-NEXT System Architecture, http://www.hbb-next.eu/index.php/documents

[20] D2.1 Usage Scenarios and Use Cases, http://www.hbb-next.eu/index.php/documents

[21] D5.3.1 Design and Protocol: Intermediate Multimodal Interface and Context Aware Recommendation Engine, http://www.hbb-next.eu/index.php/documents

[22] Ban J., Féder M., Omelina Ľ., Oravec M., Pavlovičová J. (2012). "Face Recognition Methods for Multimodal Interface", 5th joint IFIP Wireless and Mobile Networking Conference, 19-20th September 2012, Bratislava, Slovakia, ISBN: 978-1-4673-2994-1, pp. 110-113.

[23] Ban J., Féder M., Omelina Ľ., Oravec M., Pavlovičová J. (2013). "Face Recognition Under Partial Occlusion and Noise", The IEEE Eurocon 2013 Conference, 1-4th July 2013, Zagreb, Croatia,[Accepted Contribution].

[24] P. J. Phillips, H. Welchsler, J. Huang, and P. Rauss, "The FERET Database and Evaluation Procedure For Face Recognition Algorithms," Image and Vision Computing, Vol. 16, pp. 295-306, ISSN: 0262-8856, 1998.

[25] T. Sim, S. Baker, M. Bsat, The CMU Pose, Illumination, and Expression (PIE) database. Automatic Face and Gesture Recognition. pp. 46-51, 2002.

[26] M. Turk, A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, Vol. 3, No. 1, pp 71-86, 1991.

[27] B. Scholkopf, A. Smola, K. R. Muller, "Kernel Principal Component Analysis," Advances in Kernel Methods – Support Vector Learning. MIT press Cambridge MA. pp. 327-352, 1999.

[28] G. Baudt, F. Anouar, Generalized Discriminant Analysis Using a Kernel Approach. Neural Computation, 2000.

[29] K. Hlaváčková, R. Neruda, "Radial Basis Function Networks," Neural Network World, Vol.3, No.1, pp. 93-102, 1993.

[30] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning 20, pp 1–25, 1995.

[31] Ojala, T., Pietikainen, M. & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, pp. 971-987.

[32] Oravec, M., Pavlovičová, J., Mazanec, J., Omelina, Ľ., Féder, M., Ban, J. (2011). Efficiency of Recognition Methods for Single Sample per Person Based Face Recognition. Chapter in monograph Reviews, Refinements and New Ideas in Face Recognition (Ed. By Peter M. Corcoran). ISBN 978-953-307-368-2, IN-TECH, Croatia, pp. 181-206.

[33] Oravec Miloš, Mazanec Ján, Pavlovičová Jarmila, Pavel Eiben,Fedor Lehocki: Face Recognition in Ideal and Noisy Conditions Using Support Vector Machines, PCA and LDA, Face Recognition (Ed. Milos Oravec), ISBN: 978-953-307-060-5, IN-TECH, 2010.

[34] X.Tan, S.Chen, Z.-H. Zhou, and F. Zhang, Face Recognition from a Single Image per Person: A Survey, Pattern Recognition, 39(9), pp.1725-1745. 2006.

[35] R. Verschae, J. Ruiz-del-Solar, and M. Correa, "Face recognition in unconstrained environments: a comparative study," in Proceedings of the Workshop on Faces in Real-Life Images: Detection, Alignment, and Recognition (ECCV '08), pp. 1–12, Marseille, France, October 2008.

[36] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," Pattern Analysis and Machine Intelligence, 2011.

[37] http://www.nist.gov/itl/iad/ig/frvt-home.cfm

[38] J. G. Daugman, "High confidence visual recognition of persons by a test of statistical independence," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 15, no. 11, pp. 1148–1161, 1993.

[39] J. Daugman, "How Iris Recognition Works," IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, pp. 21–30, Jan. 2004.

[40] P.V.C. Hough, "Machine Analysis of Bubble Chamber Pictures", Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

[41] F. Gomez Marmol, G. Martınez Perez, "Providing Trust in Wireless Sensor Networks using a Bio-Inspired Technique", Telecommunication Systems Journal 46 (2011) 163–180.

[42] Félix Gómez Mármol, Gregorio Martínez Pérez, "State of the art in trust and reputation models in P2P networks", Handbook of Peer-to-Peer Networking, Springer, Eds: X. Shen, H. Yu, J. Buford, M. Akon, ISBN: 978-0-387-09750-3, pp 761-784, 2010

[43] M. Omar, Y. Challal, A. Bouabdallah, "Reliable and fully distributed trust model for mobile ad hoc networks", Computers and Security 28 (2009) 199–214.

[44] F. Gomez Marmol, G. Martınez Perez, A. F. Gomez Skarmeta, "TACS, a Trust Model for P2P Networks", Wireless Personal Communications, Special Issue on "Information Security and data protection in Future Generation Communication and Networking" 51 (2009) 153–164.

[45] Y. Wang, Y. Tao, P. Yu, F. Xu, J. Lu, "A Trust Evolution Model for P2P Networks", in: Autonomic and Trusted Computing, number 4610 in LNCS, 4th International Conference, ATC 2007, Springer, Hong Kong, China, 2007, pp. 216–225.

[46] C. Huang, H. Hu, Z.Wang, "A dynamic trust model based on feedback control mechanism for P2P applications", in: Autonomic and Trusted Computing, number 4158 in LNCS, Springer, Wuhan, China, 2006, pp. 312–321.

[47] S. Marti, H. Garcıa-Molina, "Identity crisis: anonymity vs reputation in P2P systems", in: Proceedings for the Third International Conference on Peer-to-Peer Computing (P2P 2003), Linkoping, Sweden, pp. 134–141.

[48] S. K. Bansal, A. Bansal, M. Blake, "Trust-based dynamic web service composition using social network analysis", in: IEEE International Workshop on Business Applications for Social Network Analysis (BASNA 2010).

[49] C.-W. Hang, M. P. Singh, "Selecting trustworthy service in service-oriented environments", in: The 12th AAMAS Workshop on Trust in Agent Societies.

[50] Z. Malik, A. Bouguettaya, "Reputation bootstrapping for trust establishment among web services", IEEE Internet Computing 13 (2009) 40–47.

[51] S. Paradesi, P. Doshi, S. Swaika, "Integrating behavioral trust in web service compositions", in: Proceedings of the 2009 IEEE International Conference on Web Services, ICWS '09, pp. 453–460.

[52] P. J. Windley, K. Tew, D. Daley, "A framework for building reputation systems", in: Proceedings of the Sixteenth International World Wide Web Conference, WWW2007, Ban, Canada, pp. 1–10.

[53] Google TV, www.google.com/tv

[54] Ben Eaton, Silvia Elaluf-calderwood, Carsten Sørensen, and Youngjin Yoo, "Dynamic Structures of Control and Generativity in Digital Ecosystem Service Innovation : The Cases of the Apple and Google Mobile App Stores". Information Systems and Innovation Group, London School of Economics and Political Science Working Paper Series, Business 44, no. 0: 0-25. April 2011

[55] Apple TV, http://www.apple.com/appletv

[56] kiTunes, http://www.apple.com/itunes

[57] Android Market, https://market.android.com

[58] Samsung Smart TV, AppStore, http://www.samsung.com/us/appstore

[59] Félix Gómez Mármol, Gregorio Martínez Pérez, "Security Threats Scenarios in Trust and Reputation Models for Distributed Systems", Elsevier Computers & Security, vol. 28, no. 7, pp. 545-556, 2009

[60] Elisabetta Carrara and Giles Hogben, "Reputation-based Systems: a security analysis", ENISA Position Paper No.2, October 2007

[61] Randy Farmer and Bryce Glass, "Building Web Reputation Systems", O'Reilly Media, March 2010

[62] D3.2 DESIGN AND PROTOCOL (High Level Architecture): User ID, Profile, Application Reputation Framework, http://www.hbb-next.eu/index.php/documents

9. Abbreviations

OASIS ORMS	OASIS Open Reputation Management Systems
AKA	Authentication and Key Agreement
AP	Aggregation Proxy
BSS	Business Support System
CDR	Charging Data Record
CF…	Call Forwarding…
CH	Communication Hold
CLF	Connectivity Session Location Repository Function
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
HLD	High Level Design
HSS	Home Subscriber Server
ILOM	Integrated Lights Out Management
LI	Lawful Intercept
LLD	Low Level Design
QoS	Quality of Service
RTP	Real-time Transport Protocol
UA	User Agent

10. Annex A

The Annex A contains examples of parametric files mentioned in the section 2.1.5.

An example of config_param.txt:

```
vzorkovacia_frekvencia 22050            #sampling frequency
dlzka_ramca 20                          #frame length
posun_ramca 10                          #frame shift
minimalna_frekvencia 250                      #min. Frequency
maximalna_frekvencia 8000               #max. frequency
pocet_filtrov 28                        #number of filters
dct_min_coef 1                          #first DCT coefficient
dct_max_coef 20                         #last DCT coefficient
prah_min_energie 0.02                   #minimal energy threshold
najmensi_absolutny_vykon 130            #minimal power
preemfaza -0.97                         #preemphase
CepstralMeanSubtraction 0               # true or false
AdaptacnyKeficientPriemeruCepstra 0.99863     #adaptation parameter for
mean
zlozka data\                            #directory of the data
```

An example of config_kd_knn.txt:

```
PocetSusedov 4                          #number of neighbours
MaxPocetVektorovNaList 160              #number of vector per leaf
VahovanyKNN 1                           #weighted or classical KNN
LokalnaVzdialenost Euclid               #local distance type
RozhodnutieVahovane 1                   #majority rule or weighted
decision
DlzkaNahravkyTrain 14                   #length of training recording
MinDlzkaNahravkyTrain 5                       #min. length of training
recording
DlzkaBuffraRozpoznavanie 700            #length of a recording buffer
PocetBuffrovRozpoznavanie 4             #number of buffers
MinPomerReci 0.5                        #min. ratio of detected speech
DatovySubor knn_vektory.knn             #file name of sample database
SuborDatabazyMien databaza_hovoriacich.txt    #file name with speaker
names
```

11. Annex B. Reviewed data model for the Trust and Reputation Module

In order to ease the communication and transfer of reputation information, both within the internal modules constituting the trust and reputation enabler and towards external modules, we have adopted a set of data structures inspired in the specification done by OASIS ORMS (Open Reputation Management Systems [15]).

11.1. Reputation bundle

The `<ReputationBundle>` element can contain one or more `<Reputation>` elements to optionally make a group of reputation instances. Within the context of HBB-NEXT, this data structure can be used, for instance, by the AppStore to retrieve the `<Reputation>` elements of a bunch of applications, all in once.

The following schema fragment defines the `<ReputationBundle>` element and its `ReputationBundleType` complex type:

```
<element name="ReputationBundle" type="ReputationBundleType"/>
<complexType name="ReputationBundleType">
<sequence>
<element ref="Issuer" minOccurs="1"/>
<element ref="Reputation" minOccurs="1" maxOccurs="unbounded"/>
</sequence>
<attribute name="id" type="String" use="optional"/>
</complexType>
```

The `<ReputationBundle>` element will be used, within the context of HBB-NEXT, to represent the reputation score given to a set of HBB-NEXT applications.

One example of use of the `<ReputationBundle>` element would be the following one:

```
<ReputationBundle id=”rpb001”>
     <Issuer id=”Alice”>
     ...
     </Issuer>
     <Reputation id=”...”>
           ...
     </Reputation>
     <Reputation id=”...”>
```

```
            ...
      </Reputation>
      ...
</ReputationBundle>
```

11.2. Reputation

The following schema fragment defines the `<Reputation>` element and its `ReputationType` complex type:

```
<element name="Reputation" type="ReputationType"/>
<complexType name="ReputationType">
<sequence>
<element ref="Subject" minOccurs="1"/>
<element ref="Score" minOccurs="1"/>
<element ref="Date" minOccurs="1"/>
<element ref="FeedbackBundle" use="optional"/>
</sequence>
<attribute name="id" type="anyURI" use="optional"/>
</complexType>
```

The `<Reputation>` element will be used, within the context of HBB-NEXT, to represent the reputation score given to a particular HBB-NEXT application. It might contain, optionally, a `<FeedbackBundle>` element containing the set of feedbacks provided by previous users regarding this specific HBB-NEXT application.

One example of use of the <Reputation> element would be the following one:

```
<Reputation id=”...”>
      <Subject>
            AngryPigeon
      </Subject>
      <Score>
            0.68
      </Score>
      <Date>
            2012-09-30T09:30:10+02:00
      </Date>
      <FeedbackBundle id=”...”>
      ...
```

```
    </FeedbackBundle>
</Reputation>
```

11.3. Subject

The `<Subject>` element contains a String value which identifies the entity evaluated by this document.

The following schema fragment defines the `<Subject>` element:

```
<element name="Subject" type="String"/>
```

The `<Subject>` element will be used, within the context of HBB-NEXT, to represent a particular HBB-NEXT application whose reputation needs to be computed or known. Moreover, it could be used as well to identify the developer or provider of a specific application requesting access to user's attributes. One example of use of the `<Subject>` element would be the following one:

```
<Subject>
    AngryPigeon
</Subject>
```

11.4. Score

The `<Score>` element contains a double value of a reputation score contained within a `<Reputation>` element.

The following schema fragment defines the `<Score>` element:

```
<element name="Score" type="double"/>
```

One example of use of the `<Score>` element would be the following one:

```
<Score>
    0.68
</Score>
```

11.5. Date

The `<Date>` element contains a time value which specifies the dates defined in the namespace of the `<Context>` element. The value MUST be expressed in UTC form and MUST NOT use fractional seconds.

The following schema fragment defines the `<Date>` element and its DateType complex type:

```
<element name="Date" type="DateType"/>
<complexType name="DateType">
<simpleContent>
<extension base="dateTime">
<attribute name="type" type="anyURI" use="required"/>
</extension>
</simpleContent>
</complexType>
```

One example of use of the `<Date>` element would be the following one:

```
<Date>
    2012-09-30T09:30:10+02:00
</Date>
```

11.6. Feedback bundle

The `<FeedbackBundle>` element can contain one or more `<Feedback>` elements to optionally make a group of feedback instances. This can be used, for instance, and within the context of HBB-NEXT, to retrieve all the feedbacks related to a concrete application in the AppStore.

The following schema fragment defines the `<FeedbackBundle>` element and its `FeedbackBundleType` complex type:

```
<element name="FeedbackBundle" type="FeedbackBundleType"/>
<complexType name="FeedbackBundleType">
<sequence>
<element ref="Subject" minOccurs="1"/>
<element ref="Feedback" minOccurs="1" maxOccurs="unbounded"/>
</sequence>
<attribute name="id" type="anyURI" use="optional"/>
</complexType>
```

The `<FeedbackBundle>` element will be used, within the context of HBB-NEXT, to represent the collection of feedbacks provided by previous users regarding a concrete HBB-NEXT application.

One example of use of the `<FeedbackBundle>` element would be the following one:

```
<FeedbackBundle id=”fbb001”>
     <Subject>
           AngryPigeon
     </Subject>
     <Feedback id=”fb001”>
     ...
     </Feedback>
     <Feedback id=”fb002”>
     ...
     </Feedback>
     ...
</FeedbackBundle>
```

11.7. Feedback

The following schema fragment defines the `<Feedback>` element and its `FeedbackType` complex type:

```
<element name="Feedback" type="FeedbackType"/>
<complexType name="FeedbackType">
<sequence>
<element ref="Issuer" minOccurs="1"/>
<element ref="Subject" minOccurs="0"/>
<element ref="Score" minOccurs="1"/>
<element ref="Date" minOccurs="1"/>
<element ref="Comment" use="optional"/>
</sequence>
<attribute name="id" type="anyURI" use="optional"/>
</complexType>
```

The `<Feedback>` element will be used, within the context of HBB-NEXT, to represent the feedback provided by a specific previous user (`<Issuer>`) regarding a concrete HBB-NEXT application (`<Subject>`).

One example of use of the `<Feedback>` element would be the following one:

```
<Feedback id=”fb001”>
     <Issuer id=”Alice”>
     ...
     </Issuer>
     <Subject>
           AngryPigeon
     </Subject>
     <Score>
           0.8
     </Score>
     <Date>
           2012-09-28T10:22:30+02:00
     </Date>
     <Comment>
           Nice application, easy to use
     </Comment>
</Feedback>
```

11.8. Issuer

The `<Issuer>` element contains a String value which identifies the evaluating entity.

The following schema fragment defines the `<Issuer>` element:

```
<element name="Issuer" type="IssuerType"/>
<complexType name="IssuerType">
<sequence>
<element ref="Preference" maxOccurs="unbounded" use="optional"/>
</sequence>
<attribute name="id" type="String"/>
</complexType>
```

The `<Issuer>` element will be used, within the context of HBB-NEXT, to represent the specific end-user providing a feedback regarding a concrete HBB-NEXT application. Moreover, this end-user will have some defined preferences used to compute a customized reputation score.

One example of use of the `<Issuer>` element would be the following one:

```
<Issuer id=”Alice”>
     <Preference name=”price”>
     ...
     </Preference>
     <Preference name=”usability”>
     ...
     </Preference>
     ...
</Issuer>
```

11.9. Comment

The `<Comment>` element contains a string value which represents the optional opinion message left by an end-user when providing feedback about a specific HBB-NEXT application.

The following schema fragment defines the `<Comment>` element:

```
<element name="Comment" type="string"/>
```

One example of use of the `<Comment>` element would be the following one:

```
<Comment>
     Nice application, easy to use
</Comment>
```

11.10. Preference

The `<Preference>` element contains a String value which identifies a concrete user’s preference regarding the applications provided in the AppStore to determine how important this feature is to the end user.

The following schema fragment defines the `<Preference>` element:

```
<element name="Preference" type="PreferenceType"/>
<complexType name="PreferenceType">
<attribute name="name" type="string"/>
<sequence>
<element ref="Score" minOccurs="1"/>
</sequence>
```

```
</complexType>
```

One example of use of the `<Preference>` element would be the following one:

```
<Preference name=”price”>
     <Score>
          0.8
     </Score>
</Preference>
```

11.11. Device capabilities

The `<DeviceCapabilities>` element is used to represent some of the features of the device used to (potentially) determine the concrete reputation computation engine to apply.

The following schema fragment defines the `<DeviceCapabilities>` element:

```
<element name="DeviceCapabilities" type="DeviceCapabilitiesType"/>
<complexType name="DeviceCapabilitiesType">
<attribute name="id" type="String"/>
<sequence>
<element ref="DeviceFeature" minOccurs="1" maxOccurs="unbounded"/>
</sequence>
</complexType>

<element name="DeviceFeature" type="DeviceFeatureType"/>
<complexType name="DeviceFeatureType">
<attribute name="name" type="String"/>
<attribute name="value" type="String"/>
</complexType>
```

One example of use of the `<DeviceCapabilities>` element would be the following one:

```
<DeviceCapabilities id=”dc001”>
     <DeviceFeature name=”screen-resolution” value=”800X600” />
     <DeviceFeature name=”user-input” value=”keyboard” />
     ...
</DeviceCapabilities>
```

11.12. System conditions

The `<SystemConditions>` element is used to represent some of the features of the environment used to (potentially) determine the concrete reputation computation engine to apply.

The following schema fragment defines the `<SystemConditions>` element:

```
<element name="SystemConditions" type="SystemConditionsType"/>
<complexType name="SystemConditionsType">
<attribute name="id" type="String"/>
<sequence>
<element ref="SystemCondition" minOccurs="1" maxOccurs="unbounded"/>
</sequence>
</complexType>

<element name="SystemCondition" type="SystemConditionType"/>
<complexType name="SystemConditionType">
<attribute name="name" type="String"/>
<attribute name="value" type="String"/>
</complexType>
```

One example of use of the `<SystemCondition>` element would be the following one:

```
<SystemCondition id=”sc001”>
     <SystemCondition name=”bandwidth” value=”1MB” />
     <SystemCondition name=”cpu-usage” value=”63%”” />
     ...
</SystemCondition>
```

11.13. Example

Next we present a comprehensive example of use of the `<ReputationBundle>` element containing all the previously shown elements:

```
<ReputationBundle id=”rpb001”>
     <Reputation id=”rep001”>
           <Issuer>
                 Alice
           </Issuer>
           <Subject>
```

```
            AngryPigeon
      </Subject>
      <Score>
            0.68
      </Score>
      <Date type=”xs:dateTime”>
            2012-09-30T09:30:10+02:00
      </Date>
      <FeedbackBundle id=”fbb001”>
            <Feedback id=”fb001”>
                  <Issuer id=”Alice”>
                        <Preference name=”price”>
                              <Score>
                                    0.4
                              </Score>
                        </Preference>
                        <Preference name=”usability”>
                              <Score>
                                    0.8
                              </Score>
                        </Preference>
                  </Issuer>
                  <Score>
                        0.8
                  </Score>
                  <Datetype=”xs:dateTime”>
                        2012-09-28T10:22:30+02:00
                  </Date>
                  <Comment>
                        Nice application, easy to use
                  </Comment>
            </Feedback>
            <Feedback id=”fb002”>
                  <Issuer id=”Bob”>
                        <Preference name=”price”>
                              <Score>
                                    0.6
                        </Score>
                  </Preference>
                  <Preference name=”usability”>
                        <Score>
                              0.6
                        </Score>
                  </Preference>
            </Issuer>
            <Score>
                  0.4
            </Score>
            <Datetype=”xs:dateTime”>
                  2012-09-27T07:26:54+02:00
            </Date>
            <Comment>
                  It becomes hard from level 15
            </Comment>
      </Feedback>
   </FeedbackBundle>
</Reputation>
<Reputation id=”rep002”>
   <Subject>
         freeMaps
   </Subject>
   <Score>
         0.8
   </Score>
   <Date type=”xs:dateTime”>
         2012-09-30T09:30:10+02:00
   </Date>
   <FeedbackBundle id=”fbb001”>
         <Feedback id=”fb011”>
               <Issuer id=”Alice”>
                     <Preference name=”price”>
                           <Score>
                                 0.4
                           </Score>
```

```
                              </Preference>
                              <Preference name=”usability”>
                                    <Score>
                                          0.8
                                    </Score>
                              </Preference>
                        </Issuer>
                        <Score>
                              0.8
                        </Score>
                        <Datetype=”xs:dateTime”>
                              2012-09-28T10:22:30+02:00
                        </Date>
                        <Comment>
                              I really recommend this app.
                        </Comment>
                  </Feedback>
            </FeedbackBundle>
      </Reputation>
</ReputationBundle>
```

In this example the `<ReputationBundle>` element contains, in turn, two `<Reputation>` elements. The first one contains the reputation of an application named AngryPigeon (within `<Subject>`). The reputation value is represented as a double value between 0 and 1, which in this case is 0.68 (specified in `<Score>`). This element also specifies the date where the reputation has been computed (`<Date>`).

In addition to that, the `<Reputation>` element contains a `<FeedbackBundle>` element, which includes set of <Feedback> elements. In this case, there are two `<Feedback>` elements. The first one describes the feedback which the user identified as Alice has given about AngryPigeon. It specifies the score she gives to the application (within the `<Score>` element) date when she provides the feedback (`<Date>` element), some additional comments that Alice provided (`<Comment>` element) and it also includes a set of parameters defining Alice’s preferences (within the `<Preference>` element).

The `<Preference>` element describes the importance that a user gives to a certain parameter. In this example, there are defined two parameters, namely price and usability.

The former has a relevance of 0.4 for Alice, on a scale between 0 and 1, while the latter has a relevance of 0.8 for Alice.

The second `<Feedback>` element related to the application AngryPigeon describes the feedback given by Bob. Bob scored the application with a 0.4. After taking into account these feedbacks the reputation computed to the application AngryPigeon is 0.68.

In a similar way, the other `<Reputation>` element describes the reputation of an application named freeMaps. This application has a score of 0.8 out of 1, and contains a `<Feedback>` element representing the feedback given by Alice to this application.