

Sound And Music for Everyone Everyday Everywhere PUBLIC  
Every way

Editors: Maurizio Mancini (UGDIST), Giovanna Varni (UGDIST), 31 August 2009  
Antonio Camurri (UGDIST), Gualtiero Volpe (UGDIST), Antti Eronen  
(TKK), Jari Kleimola (TKK)

# SAME

## D6.3 – STANDARDIZATION PLAN

<i>Version</i>	<i>Edited by</i>	
V1.0	UGDIST	Full version
V0.5	UGDIST	Preliminary version
V0.1	UGDIST	First draft

Sound And Music for Everyone Everyday Everywhere  
Every way

PUBLIC

Editors: Maurizio Mancini (UGDIST), Giovanna Varni (UGDIST),  
Antonio Camurri (UGDIST), Gualtiero Volpe (UGDIST), Antti Eronen  
(TKK), Jari Kleimola (TKK) 31 August 2009

## TABLE OF CONTENTS

Table of contents.....	2
1 Introduction.....	3
2 Terms, acronyms, abbreviations.....	3
3 existing standards.....	4
4 Proposed standards.....	5
4.1 Gesture descriptors.....	5
4.2 Low level features.....	16
4.3 Audio descriptors.....	25
5 References.....	29

## **1 INTRODUCTION**

This document describes how the SAME project could contribute to future standards in user centric media and future internet applications. In particular, this Deliverable presents some of the main innovative contributes of the SAME project in terms of data and protocols. The SAME project focuses on contributes on expressive, social, context, audio data containers, on audio and metadata streaming, and on mobile platform.

These data containers establish in the project a shared standard for developing applications running on the SAME platform, that is, involving client/server data exchange and audio streaming, where clients are mobile devices like PDAs and phones and the servers follow the SAME platform specification.

Some of these structures have already been implemented in the first SAME applications described in Deliverable 5.1.

## **2 TERMS, ACRONYMS, ABBREVIATIONS**

E2E	End-to-End SAME Platform described in Deliverable 2.3
HTTP	Hypertext Transfer Protocol
PDA	Personal Digital Assistant, an handheld mobile computer, or PalmTop
SAME	Sound And Music for Everyone Everyday Everywhere Every way, EU-ICT FP7 project
VST	Virtual Studio Technology by Steinberg is an interface for integrating software audio synthesizer and effect plugins with audio editors and hard-disk recording systems.

### 3 EXISTING STANDARDS

The SAME E2E platform is based on a number of existing standard platforms and protocols allowing one to build applications that measure, send and receive data from physical sensors. We report them in Table 1 and then we provide a brief description of each of them.

Platform/Protocol	Type
C++	platform
EyesWeb	platform
PD	platform
Python	platform
S60	platform
Bluetooth	standard
GPS	standard
MIDI	standard
OSC	standard
TCP	standard
UDP	standard
WLAN	standard

Table 1: List of the existing standards used in SAME

Platforms:

- C++: object oriented multi-platform compiled programming language; in SAME it is used to program applications on both the E2E platform and the mobile devices
- EyesWeb: open software platform to support the development of real-time multimodal distributed interactive applications
- PD: (Pure Data) is a real-time graphical programming environment for audio, video, and graphical processing
- Python: object oriented multi-platform interpreted programming language; in SAME it is used to implement application for the mobile devices
- S60: software platform for mobile phones running the Symbian Operating System; it is installed on many Nokia Smart Phones and on

other manufacturers phones (Lenovo, LG Electronics, Panasonic and Samsung)

Protocols:

- Bluetooth: wireless protocol for connecting portable and fixed devices at short distances
- GPS: (Global Positioning System) satellite based navigation system allowing to determine 3D position (latitude, longitude, altitude) everywhere on Earth
- MIDI: (Musical Instrument Digital Interface) standard protocol for communicating between music and audio devices
- OSC: (Open Sound Control) networked based protocol candidate for extending current MIDI limitations
- TCP: (Transmission Control Protocol) communication protocol over the Internet for data exchange between programs running on separate computers
- UDP: (User Datagram Protocol) unreliable communication protocol not based on an established connection but on the transmission of unordered datagrams between Internet connected hosts
- WLAN: Wireless Local Area Networking protocol

## **4 PROPOSED STANDARDS**

### **4.1 Gesture descriptors**

Expressive gesture descriptors are mainly defined in Deliverable 4.1 and some of them are used in the applications presented in Deliverable 5.1; they are used to describe gesture expressive characteristics at low (e.g., amplitude, speed, symmetry) and high (e.g., impulsivity, fluidity) level.

These gesture descriptors could be considered for submission to future MPEG initiatives, in the same way as MPEG7 deals with the description of

audio and music information. Table 2 resumes the gesture descriptors, for each one of them we provide the following information:

1. Descriptor name;
2. The existing platform and/or standard on which the descriptor is based;
3. the deliverable Section number providing a definition of the descriptor;
4. the implementation status of the descriptor;
5. the descriptor level;
6. the value in the Gesture Descriptor Frame representing the descriptor (see next Section);

Descriptor name	Based on existing standard/platform	Described in deliverable	Implemented	Low/high	Value in GDF
Motion Index	EyesWeb, OSC	D4.1-S3.1.1 D5.1	yes	low	$g_1$
Internal Motion Index	EyesWeb, OSC	D4.1-S3.1.3	yes	low	$g_2$
Contraction Index	EyesWeb, OSC	D4.1-S3.1.4	yes	low	$g_3$
Translational Motion Index	EyesWeb	D4.1-S3.1.2	yes	low	$g_4$
Symmetry Index	EyesWeb	D4.1-S3.1.5	yes	low	$g_5$
Impulsivity Index	EyesWeb	D4.1-S3.1.6	yes	high	$g_6$
Fluidity Index	EyesWeb	D4.1-S3.1.8	yes	high	$g_7$
Directness Index	EyesWeb	D4.1-S3.1.11	yes	high	$g_8$
Periodicity Index	EyesWeb	D4.1-S3.1.7	no	high	$g_9$
Space Occupation Area	EyesWeb	D4.1-S3.1.10	no	high	$g_{10}$
Occupation Rates <sup>1</sup>	EyesWeb	D4.1-S3.1.13	no	high	$g_{11}+KDF$
Kinematical measures on trajectories <sup>1</sup>	EyesWeb, OSC	D4.1-S3.1.9 D5.1	yes	low	$g_{11}+KDF$

Table 2: low and high level gesture descriptors.

<sup>1</sup> Kinematical measures and Occupation rates are codified separately respect to all the other descriptors; see Section 4.1.2, 4.1.13 and 4.1.14.

### 4.1.1 Gesture Description Frame (GDF)

Our proposal for standardization consists in creating a representation that meets the following requirements:

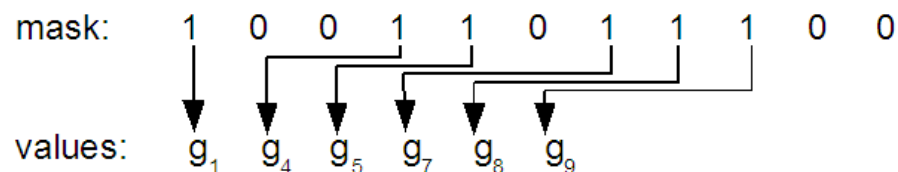
- data must be integer;
- data must be dimension less;
- data size must be optimized;

What we propose is to implement a Gesture Descriptor Frame (GDF), consisting of two elements: a *masking* vector and a *values* vector. The masking vector is a sequence of 11 elements which can be either 0 or 1 corresponding to the current number of gesture descriptors included in SAME; however in the future this number could be changed according to the development of other descriptors.

The values vector is a sequence of 11 integers  $\langle g_1, \dots, g_{11} \rangle$  in the range  $[0, 65535]$  which represents the gesture descriptors value in Table 3; again, the number of elements of this vector could change if other gesture descriptors will be developed.

For each element in the mask, if the element is 0 then the corresponding element in the value vector is not present; if it is 1 then the corresponding element in the value vector is present, except for the last value  $g_{11}$ , as we describe in Section 4.1.2, 4.1.13 and 4.1.14.

Here is an example of GDF:



in the above example 6 descriptors out of the 11 available are enabled, thus only their values are sent, optimizing bandwidth.

By defining a single GDF we can describe the dynamic evolution of a gesture by sending a stream of GDFs, each frame identified by a time label. At the implementation level GDFs could be encoded in OSC messages to be transmitted between the E2E platform components. The GDF format is similar to the GDIF format introduced by Jensenius et al. in [5]. GDIF aims at establishing a standard description format for

several aspects of a gesture: its low level physical data, descriptive information and high level features like velocity or intensity. In a similar way we propose to integrate low and high level descriptors in the same framework. Differences with GDIF are at transmission level: GDIF format consists of several synchronized streams that are transmitted in parallel; instead, we propose a variable length frame format that do not impose to establish network connections to stream data: in our format unavailable information is simply omitted and available information is packed and transmitted as a single frame in asynchronous way. Given these similarities and differences we could also propose an integration of the two formats by extending the existing GDIF format to include also our gesture descriptors by adding new streams of data to the existing ones. However such implementation could be inappropriate for a distributed architecture like the one in SAME. In the following Sections we describe the 11 gesture features, for each of them we provide a brief description and its possible values.

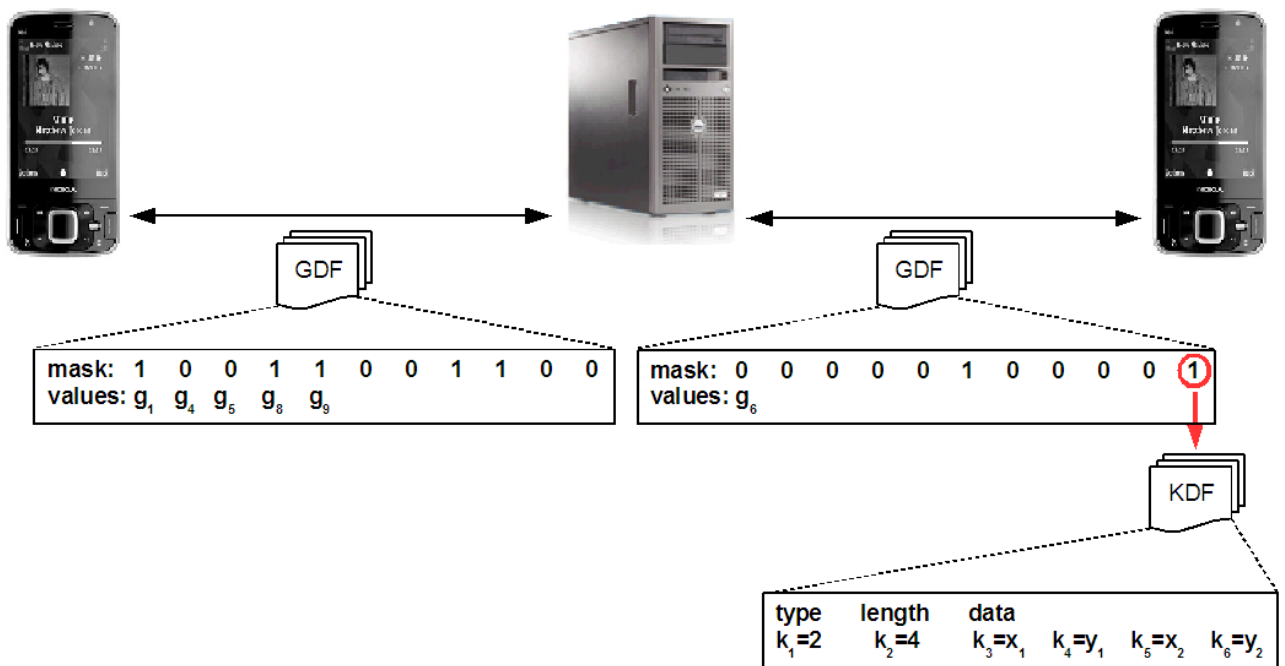
#### 4.1.2 Kinematic Description Frame (KDF)

If in GDF the value of  $g_{11}$  is 1, it means that the descriptor represents the kinematical features of a trajectory: e.g., the position of the silhouette barycenter, the coordinates of a bounding box and so on. In this case, to represent such information, we propose to implement a Kinematic Description Frame (KDF), a variable length integer vector  $\langle k_1, \dots, k_n \rangle$  in which:

- $k_1$  identifies the type of the KDF:
  1. barycenter coordinates;
  2. bounding box coordinates;
  3. convex hull coordinates;
  4. occupation rates;
  5. trajectory point coordinates;
  6. ...
- $k_2$  stores the length of the KDF data; for example a Rectangle of Interest consists of 2 points, that is 4 values, while a

Convex Hull can be composed by an arbitrary number of points;

- $k_3, \dots, k_n$ : the coordinates of the type specified by  $k_1$ ;



### 4.1.3 Motion Index

The Motion Index (MI, formerly Quantity of Motion, QoM) is computed as the area (i.e., number of pixels) of a Silhouette Motion Image, an image carrying information about variations of the silhouette shape and position in the last few frames.

Feature	low values (~0)	high values (~65535)
Motion Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_1 = 1</math></li> <li>• <math>v_1 = \text{Motion Index}</math></li> </ul>	no movement, e.g., silhouette is constant between two consecutive frames.	maximal movement, e.g., the silhouette covering the entire video frame has moved out of the field of view.

#### 4.1.4 Internal Motion Index

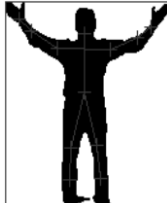

The Internal Motion Index consists of a measure of the amount of motion detected inside a blob silhouette. The darker area in the Figure below shows the Internal Motion of the silhouette:



Feature	low values (~0)	high values (~65535)
Internal Motion Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_2 = 1</math></li> <li>• <math>v_2 = \text{Internal Motion Index}</math></li> </ul>	no motion or, e.g., limbs are moving outside the rest of the body.	high movement activity in front of the body (i.e., limbs occluding trunk).

#### 4.1.5 Contraction Index

Contraction Index (CI) is a measure, ranging from 0 to 65535, of how a subject uses the space surrounding her in terms of contraction/expansion of the body with respect to its centre of gravity.

Feature	low values (~0)	high values (~65535)
Contraction Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_3 = 1</math></li> <li>• <math>v_3 = \text{Contraction Index}</math></li> </ul>	very expanded subject, e.g., user open his arms and legs. 	very contracted subject. 

### 4.1.6 Translational Motion Index

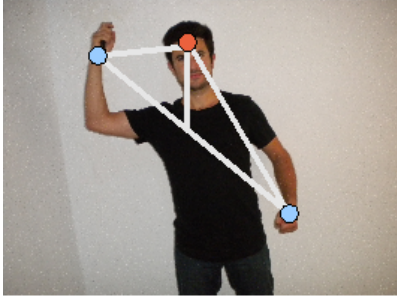
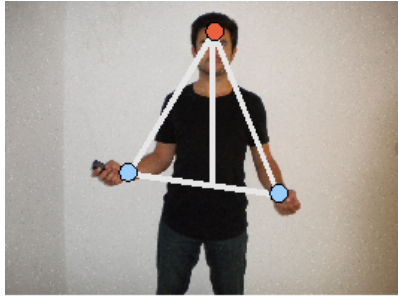
The Translational Motion Index (TMI) is a Motion Index computed on translational movements only, that is, it is computed as:

$$TMI = MI - IMI$$

Feature	low values (~0)	high values (~65535)
Translational Motion Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_4 = 1</math></li> <li>• <math>v_4 =</math> Translational Motion Index</li> </ul>	e.g., the user moves her limbs but does not change her position in the space.	e.g., the user wals in front of the camera from left to right.

### 4.1.7 Symmetry Index

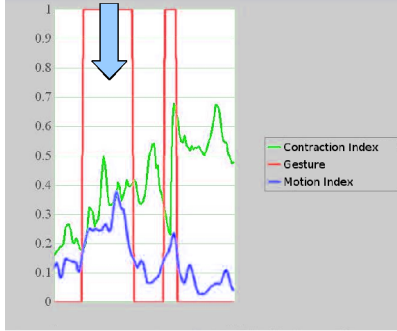
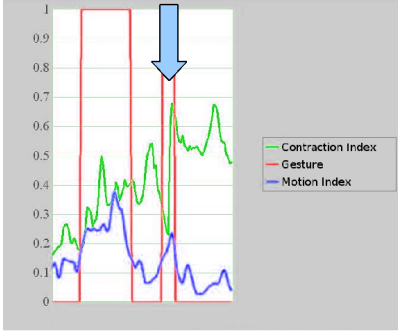
Symmetry Index (SyI) can be computed either on the bounding rectangle or directly on the blob silhouette. If the bounding rectangle is used, SyI is computed from the position of the barycenter and the left and right edges of the bounding rectangle.

Feature	low values (~0)	high values (~65535)
Symmetry Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_5 = 1</math></li> <li>• <math>v_5 =</math> Symmetry Index</li> </ul>		

### 4.1.8 Impulsivity Index

The Impulsivity Index (II) can be measured as a combination of other descriptors. First the movement energy MI is used to identify the gesture time duration  $\Delta t$ : for example, using an adaptive threshold, when the MI value become greater than the threshold we identify the gesture

beginning time; when the MI goes below the threshold we identify the ending time. In the gesture time interval we evaluate the variation of the posture by looking at the Contraction Index value. To resume, the II can be formalized as the ratio between  $\Delta CI$  and  $\Delta t$ .

Feature	no impulse (0)	impulsive (65535)
Impulsivity Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_6 = 1</math></li> <li>• <math>v_6 = \text{Impulsivity Index}</math></li> </ul>	 <p>the identified gesture in the red square has a limited variation of CI in a "long" time interval so it is not impulsive.</p>	 <p>the identified gesture in the red square has a high variation of CI in a "short" time interval so it is impulsive.</p>

#### 4.1.9 Fluidity Index

Research in [1] demonstrates a correspondence between (i) smooth trajectories performed by human arms, (ii) minimization of the third-order derivative of the hand position (called jerk in physics) and (iii) correlation between hand trajectory curvature and velocity. We use an approach similar to (iii) to determine if a trajectory is smooth or not starting from the trajectory curvature and velocity.

Feature	low values (~0)	high values (~65535)
Fluidity Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_7 = 1</math></li> <li>• <math>v_7 = \text{Fluidity Index}</math></li> </ul>		

#### 4.1.10 Directness Index

Directness Index is a measure related to the geometric shape of a movement trajectory. It provides information on how much the trajectory followed by a gesture is straight.

Feature	low values (~0)	high values (~65535)
Directness Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_8 = 1</math></li> <li>• <math>v_8 = \text{Directness Index}</math></li> </ul>		

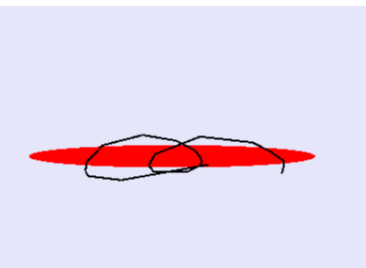
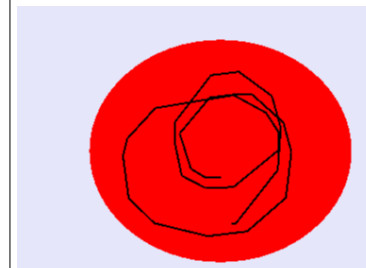
#### 4.1.11 Periodicity Index

The Periodicity Index (PI) provides information about detection of periodic movements and can be considered as a first step toward analysis of rhythm.

Feature	low values (~0)	high values (~65535)
Periodicity Index Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_9 = 1</math></li> <li>• <math>v_9 =</math> Periodicity Index</li> </ul>	user is performing random movements, there is no recurrence in the path described by her hands/body.	user movements are repetitive, i.e, there is recognizable pattern that is repeated in a consecutive way.

#### 4.1.12 Space Occupation Area

SOA is computed starting from the movement trajectory integrated over time. In such a way a bitmap is obtained, summarizing the trajectory followed along a considered time window (3 s) or along a selected gesture. An elliptical approximation of the shape of the trajectory is then computed. The area of such ellipse is taken as the Space Occupation Area.

Feature	low values (~0)	high values (~65535)
Space Occupation Area Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_{10} = 1</math></li> <li>• <math>v_{10} =</math> Space Occupation Area</li> </ul>		

#### 4.1.13 Kinematical measures on trajectories

These include kinematical measures computed on trajectories of points associated to different body parts, e.g., body centroids, barycentre,

barycentre of a specific body part (e.g., hands, feet, head), vertexes of the bounding rectangle, of the convex hull, of the triangle formed by hands and head.



To represent this data we propose the implementation of the KDF (Section 4.1.13), so the meaning of the values contained in this structure varies depending on the information represented, for example:

- barycenter coordinates: the KDF contains 2 values  $(x,y)$  in  $[0, 65535]$ , where  $(0,0)$  is the up-left corner of the video frame and  $(65535,65535)$  is the bottom-right corner;
- bounding box coordinates: the KDF contains 4 values  $(x_1,y_1,x_2,y_2)$  in  $[0, 65535]$ , where  $(x_1,y_1)$  are the coordinates of the top-left vertex of the bounding box and  $(x_2,y_2)$  is the bottom-right corner;
- convex hull coordinates: the KDF contains a sequence of  $n$  vertices  $(x_1,y_1,x_2,y_2, \dots, x_n,y_n)$ , where for every  $i$  in  $[1, \dots, n]$   $(x_i,y_i)$  is the  $i$ -th vertex of the convex hull;

#### 4.1.14 Occupation Rates

Occupation Rates are a measure of how much a motion trajectory occupied single, specific areas in the space. Given  $n$  areas in space, an Occupation Rate is associated to each area and is stored in a value of a KDF. For every  $i$  in  $[1, \dots, n]$ , if  $A_i$  is the  $i$ -th area, Occupation rate is defined as:

$$k_i = OR_i = (\text{number of trajectory coordinates in } A_i) / (\text{trajectory length})$$

Feature	low values (~0)	high values (~65535)
Occupation Rates Value in GDF: <ul style="list-style-type: none"> <li>• <math>g_{11}=1</math></li> </ul> Value in KDF: <ul style="list-style-type: none"> <li>• <math>k_1 = 4</math></li> <li>• <math>k_2 = \#</math> of areas</li> <li>• <math>k_3, \dots, k_n =</math> areas Occupation Rates</li> </ul>		

## 4.2 Low level features

Here we list the low level features that could be proposed for standardization. In our summary we include types that could serve as a descriptor for low level data such as audio, video and sensors data. All of these features are commonly present on mobile telephones and PSAs. Some of them have already been implemented in prototype applications described in Deliverable 2.2 and 5.1, and are especially dedicated to context awareness (for example, from the light sensor we could infer information about the environment).

We base the representation of the descriptors as OSC messages, that is, alphanumeric messages containing both a description, e.g., *acceleration*, and a numeric or alphabetic value, e.g., *%ax* (acceleration on the x axis). Moreover, we propose to separate control commands from data flow using different ports for receiving OSC packets and we introduce two namespaces corresponding to these channels:

- */control* for sending and receiving control commands;
- */data* for sending and receiving data;

The low level features defined in SAME are resumed in Table 2.

For each feature we report: its name (column 1); the list of existing standards and platforms which are needed to implement it (column 2); the list of Deliverable and Sections in which it is described (column 3); if the low level feature is a control or a data feature (columns 4 and 5); the status of the feature implementation and use in SAME (column 6).

In the next Section we briefly resume the description of each feature and we propose a possible implementation of a standard representation for each one of them. We also give a description of those features that we propose to include in future standardization but that are not yet implemented in the SAME platform.

Low level features					
Feature name	Based on existing standard/platform	Described in deliverable	Control	Data	Implemented
Audio commands	OSC, UDP	D5.1	*		yes
Optical flow	OSC, UDP, C++, S60	D5.1		*	yes
Accelerometer values	Python, OSC, UDP	D2.2-2.1 D5.1		*	yes
Mobile camera	C++, Bitmap, UDP	D2.2-2.6 D5.1		*	yes
MP3/Wave audio, microphone	HTTP, Shoutcast, TCP	D2.2-2.10 D5.1		*	yes
Keyboard/touch screen	OSC, UDP	D2.2-2.15 D5.1	*	*	yes
Active application	S60	D2.2-2.2	*		no
Ambient light intensity	S60	D2.2-2.3		*	no
Bluetooth / WLAN	S60, Python for S60	D2.2-2.4 D2.2-2.16	*	*	no
Calendar events	S60, Python for S60	D2.2-2.5	*		yes (REST over HTTP)
Compass data	S60, Python for S60	D2.2-2.7		*	no
GPS data	S60, Python for S60	D2.2-2.8	*	*	yes (REST over HTTP)
Inactivity time	S60, Python for S60	D2.2-2.9		*	yes (REST over HTTP)
Phone profile	S60, Python for S60	D2.2-2.12	*		yes (REST over HTTP)
Temperature	No S60 API support	D2.2-2.1		*	no

Table 3: the list of the low level proposed standard

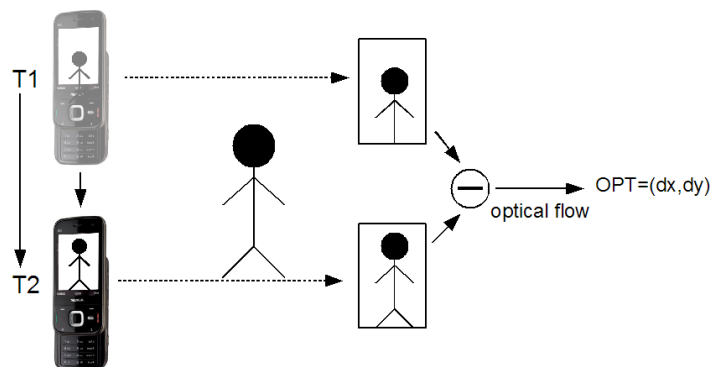
## 4.2.1 Audio commands

Feature	Control/Data	OSC messages	Message description
Audio commands	Control	/control/midi/play %c /control/midi/speed %s /control/midi/volume %v /control/midi/pitch %p	This feature controls audio and midi playback on the mobile device. Speed, volume and pitch are controlled by sending the corresponding command together

		<pre> /control/midi/repeat %c /control/audio/play %c /control/audio/speed %s /control/audio/volume %v /control/audio/repeat %c </pre>	with the value to which set the parameter to. Instead the play and repeat commands have a boolean flag to allow one to, for example, start/stop playback.
--	--	---	---

### 4.2.2 Optical flow

PDAs and Nokia mobile phones running S60 applications are equipped with a high definition and a low definition camera, capturing images at around 15 fps. By computing optical flow between two consecutive frames, one can estimate the mobile direction and quantity of movement along two axis, as illustrated in the Figure:



*Figure 1: by computing the mean optical flow between two consecutive camera snapshots we can estimate the mobile phone movement direction and magnitude.*

By comparing the pixels of two frames taken at times T1 and T2 we obtain a movement matrix in which the speed of movement of every pixel on x and y axis is reported. If we compute the mean value of these speeds we obtain an estimation of the movement of the phone along x and y. There are two possible implementations of such computation: (i) we stream the image from the phone camera to the SAME platform and we compute optical flow there; in this case we propose a standard for streaming images in Section 4.2.4; (ii) we compute the optical flow internally to the mobile phone and then we send the result to the SAME platform; in this case we propose the following format:

Feature	Control/Data	OSC messages	Message description
Optical flow	Data	/data/opt %dx %dy	The mean Optical Flow is computed internally by the PDA/mobile phone, and sent as a sequence of OSC packets to the platform or another mobile. Every message contains both the mean velocity on the X and on the Y axis.

### 4.2.3 Accelerometer values

Accelerometers embedded in PDAs and mobile phones are able to measure 3D acceleration in the range  $[-2g, 2g]$  at 30 Hz. Currently, on the implemented SAME applications, accelerometer data is readable using Python scripts and C++. Pre-processing, e.g., filtering out g and/or noise, is possible within the mobile itself or after receiving data in the application running on the E2E platform.

We defined the following OSC packet format to exchange accelerometers data between mobiles and applications.

Feature	Control/Data	OSC messages	Message description
Accelerometer values	Data	/data/acc/raw %x %y %z /data/acc/norm %x %y %z /data/acc/no-g %x %y %z	Acceleration read from the accelerometer sensor is sent as raw values or after normalizing values in the range $[-1, 1]$ . Also the g filtering could be performed before transmitting the accelerometers values.

### 4.2.4 Mobile camera

Portable devices are in most cases equipped with one or two cameras, able to take snapshots and videos. In some preliminary applications developed in SAME (see Deliverable 5.1) we implemented a protocol to send a stream of video frames through UDP. This technique allows one to send and receive low resolution black and white videos at 15 fps over a local wireless network. On a Nokia mobile phone we developed a C++ program able to capture camera frames, encode them in the EyesWeb

UDP format for images and send it through the network in a single UDP packet. As it is received in EyesWeb the image is extracted from the packet with existing algorithms and processed for example to computed Optical Flow in the server platform.

A UDP datagram has the following format:

```
struct DATAGRAM_DATATYPE_PACKET
{
    __int64 packet_ndx;
    size_t start_pos;
    size_t size;
    BYTE data[MAX_DATA_SIZE];
    size_t uncompressed_data_size;
    bool end_of_data;
};
```

The maximum size of the whole datagram is 8000 bytes and we aim to send one mobile camera frame with a single packet.

The format of an image frame header in EyesWeb is the following:

```
struct DATATYPE_HEADER
{
    TIME step_time;
    TIME creation_time;
    TIME presentation_time;
    TIME media_time;
    TIME duration;
    TIME media_duration;

    __int64 buffer_size;
};
```

After the header, by using images with 64x64 pixels gray scale we have to transmit a data buffer of:

$$64 * 64 = 4096 \text{ bytes}$$

That is, with the above format we are able to fit a whole image frame into a single UDP datagram.

#### 4.2.5 MP3/Wave audio, microphone

In the current version of the SAME platforms described in Deliverables 2.3 and 5.1 we are able to establish audio streams between the platform server and n mobile devices. In this way the same audio source that is streamed by the server can be played back on any connected mobile. Audio can be read from a wave or an MP3 file stored in the server. The format of such streaming process is SHOUTcast, a free standard defined by NullSoft and implemented as a variant of the HTTP protocol. For future development of the SAME platform we also aim to provide streaming of the audio captured by the mobile microphone.

Feature	Control/Data	SHOUTcast	Description
MP3/Wave audio, microphone	Data	Establish HTTP connection and send/receive audio stream	

#### 4.2.6 Keyboard/Touch screen

This is both a Control and a Data feature. If for example we transmit raw touch screen clicking position we are sending data, that is, x and y raw coordinates that will be interpreted in some way by the receiving application. If, instead, in the mobile device we detected a particular event when touching the screen, for example the activation of a button, then we are sending this event, that is, control information. In the same way, if on the mobile phone we press a physical keyboard button we propose to send this information as a control event. Currently only raw x and y position data is implemented in the first demo applications in SAME.

Feature	Control/Data	OSC messages	Message description
Keyboard/Touch screen	Control and/or Data	/data/touch %x %y /control/button %ID /control/key %k %b	Only raw touch screen coordinates can be sent as a data stream. Or, user events like pressing an application button or a physical key can be sent as control events.

#### 4.2.7 Active application

PDAs and Nokia S60 mobile phones are able to run several applications simultaneously: for example the user can check her daily schedule while switching to a different MP3 playing in the background. The user can start applications, switch between running applications and close applications.

Feature	Control/Data	OSC messages	Message description
Active application	Control	/control/app/switch %appID /control/app/start %appID /control/app/end %appID	Sent when the user switch to, starts or ends an application. %appID could be taken from a list of standardized applications(e.g., organizer, calculator, audio player and so on)

#### 4.2.8 Ambient light intensity

Some Nokia phones are equipped with a sensor for measuring ambient light and adapting the display backlight intensity. The information extracted by the sensor could be useful to determine whether the user moves for example from indoor to outdoor.

Feature	Control/Data	Proposed OSC messages	Message description
Ambient light intensity	Data	/data/ambient/light %value	The message is used to send current environment light condition.

#### 4.2.9 Bluetooth / WLAN

Many PDAs and phones have one or booth of these connectivity capabilities. That is, they are capable of detecting other mobile or fixed devices operating in the same area, e.g., room, building and so on. This feature is used to send control messages to the mobile device to scan the local area for other devices with wireless capabilities to, for example,

infer contextual information. Then the mobile device could send data messages to list the found devices.

Feature	Control/Data	OSC messages	Message description
Bluetooth / WLAN	Control and/or Data	/control/bt/scan /control/bt/setvisible %flag /data/bt/detected %hostname /control/wlan/scan /control/wlan/setvisible %flag /data/wlan/detected %hostname	Control messages are sent to the mobile to scan the local area for all the visible Bluetooth or WLAN hosts in the area or to show/hide the mobile device to the other devices. Data messages contains the names of the other hosts found in the local area.

#### 4.2.10 Calendar events

PDA's and S60 phones have organizer/scheduler applications in which the user can mark important dates, meetings and memo. Every event has a starting and ending time and an alarm can be defined to remind the user about the event at a given (even repetitive) time before the beginning of the event.

Feature	Control/Data	OSC messages	Message description
Calendar events	Control	/control/event/start %ectName /control/event/in %evtName /control/event/end %evtName /control/event/near %evtName %evtTime /control/event/past %evtName	These control messages are triggered when the mobile device scheduler application detects that particular conditions about scheduled events are met. For example if the starting time of a meeting is reached the mobile sends the corresponding message containing the name of the triggered event.

#### 4.2.11 Compass data

A few phone models (e.g., 6210) have an embedded compass sensor, to detect the mobile orientation relative to the Earth magnetic North Pole. Together with the GPS receiver this sensor could be used to detect user direction changes.

Feature	Control/Data	OSC messages	Message description
Compass data	Data	/data/compass/dir %degrees	These type of message contains the current orientation of the user respect to the Earth magnetic North Pole.

#### 4.2.12 GPS data

Some PDAs and mobile phones are equipped with an internal GPS antenna. Only in outdoor situations this sensor can determine the mobile position with great accuracy in real time. The possible unavailability of such feature, due to ambient conditions, should be checked before trying to access the sensor data. From GPS position we can compute the mobile altitude, speed of movement, direction of movement and accuracy of all of these features.

Feature	Control/Data	OSC messages	Message description
GPS data	Data and/or Control	/data/gps/status %f /data/gps/pos %long %lat %alt	The GPS position (latitute and longitude) and altitude are send as a data packet. If they are not available the status control packet returns false.

#### 4.2.13 Inactivity time

This feature allows one to obtain information about the total amount of user inactivity on the PDA or phone, allowing one to infer for example the level of interest/involvement the user has towards phone applications and content.

Feature	Control/Data	OSC messages	Message description
Inactivity time	Data	/data/user/inactive %delay	Data packet used to retrieve the user consecutive inactivity time.

#### 4.2.14 Phone profile

Every Nokia phone can be set by the user or automatically switch to a particular operating profile. Profiles usually are associated to context, for example *Silent mode* or *Noisy environment*. By monitoring the profile changes on the mobile phone we could extract information about for example the user social context.

Feature	Control/Data	OSC messages	Message description
Phone profile	Control	/control/user/profile/get %profID /control/user/profile/set %profID	These packet are used to retrieve of set a particular user profile.

#### 4.2.15 Temperature

Some PDAs and Nokia phones have a temperature sensor (dedicated only to battery temperature on some models) which measures or approximates the environment temperature. This feature can be used in conjunction with the ambient light sensor to get a description of the physical characteristics of the environment.

Feature	Control/Data	OSC messages	Message description
Temperature	Data	/data/ambient/temperature %value	Data message to retrieve the temperature measured by the sensor.

### 4.3 Audio descriptors

Here we introduce a hierarchical OSC namespace for accessing the synthesis parameters of a sound synthesizer. We propose also a straight-forward way to wrap the MIDI protocol as an OSC content format.

Audio descriptors				
Feature name	Based on existing standard/platform	Described in deliverable	Section number	Implemented in D3.2

tree-like addressing	OSC			yes
meta descriptors	OSC			partly
perceptual descriptors	OSC, MPEG-7	D3.1	3.1.1 – 3.3.5	partly
low level descriptors	OSC			partly
MIDI over OSC	OSC, MIDI			partly

### 4.3.1 Tree-like addressing

All synthesizer related descriptors and messages can optionally be prefixed by a two-part header (e.g., `/mob/dpw10`), which uniquely identifies the synthesizer (`/mob`) and the instrument that is implemented by the synthesizer (`/dpw10`). For example, to set the brightness attribute of a synthesized sound, the controller would send a message `'/mob/dpw10/brightness 0.75'`. The first part of the prefix is optional, because it is implied by the IP address + port pair of the OSC endpoint. The latter part is optional for monotimbral synthesis engines.

Furthermore, it might also be beneficial to group the related unit generators or synthesis parameters together, and extend the prefix depth accordingly. These subsections, their subsection and descriptors can then be arranged in a tree-form hierarchy (e.g., `'/mob/dpw10/osc1/frequency 220.0'`). The subsections can also provide units of persistence - the highest level would constitute the entire timbre in form of a patch.

Feature	Proposed OSC messages	Message description
prefix	<code>[/synth]</code>	<code>/synth</code> identifies the synthesizer. datatype string.
	<code>[/synth] [/instrument]</code>	<code>/instrument</code> identifies a multi-timbral synthesizer part (equivalent of a MIDI channel). datatype string.
	<code>[/synth] [/instrument][subsection]</code>	<code>/subsection</code> is a placeholder for a group of related descriptors or unit generators (e.g., envelope generator, FM operator).

### 4.3.2 Meta descriptors

Meta descriptors behave as macros or bundles by aggregating a group of lower level descriptors together. For example, playing a single note on a synthesizer requires the setting of several attributes at an instance (i.e., pitch, loudness and gating). Meta descriptors can also be utilized to group several synthesis parameters under a single point of control.

The benefit of meta descriptors is efficiency (the message needs to be transmitted and parsed only once), and expressivity (several parameters can be controlled at the same time).

Feature	Proposed OSC messages	Message description
meta descriptors	<pre>/note %pitch %loudness [%duration]</pre> <pre>/brightness %value</pre>	<p><i>/note</i> aggregate corresponds to the MIDI note on/off message (off implied by setting %loudness to zero). All arguments are floats.</p> <p><i>/brightness</i> is used here as an example aggregate descriptor. %value argument is either unipolar or bipolar float, and scaled in between [0..1] or [-1..1], respectively.</p>

### 4.3.3 Perceptual descriptors

Pitch, loudness, timbre and duration are terms that are utilized often when describing the characteristics of audio tones. The MPEG-7 [3] standard attempts to express these subjective and ambiguous terms as well-defined measurable quantities, by introducing the concept of audio descriptors. The MPEG-7 descriptors would in this namespace fit into the higher level concepts, and be addressable straight after the prefix (e.g., /mob/dpw10/pitch 45.3). However, because the synthesis engine does not speak MPEG-7, most descriptors need to be transformed into the synthesizer parameter space. This can be done in python.

Feature	Proposed OSC messages	Message description
perceptual descriptors	<pre>/pitch %index</pre> <pre>/loudness %value</pre>	<p>MPEG-7 scale descriptor index (float). Fractional part interpolates logarithmically between the predefined scale points.</p>

	<p>/timbre %value</p> <p>/duration %value</p>	<p>/loudness is related to the amplitude of the sound. This proposal does not attempt to define a universally valid mapping between the control and perceptual spaces: all that is stated here is that the range of the %value attribute goes from silence to the maximum level, and is scaled in between [0..1], logarithmically,</p> <p>/timbre is a placeholder for a meta descriptor, MPEG-7 InstrumentTimbre set, or a low level descriptor discussed below.</p> <p>/duration defines the length of the gating signal associated with a control event. %value is given in milliseconds.</p>
--	---	--

#### 4.3.4 Low level descriptors

Low level descriptors are synthesis technique dependent parameters. For example, a table lookup oscillator might have descriptors for the fundamental frequency, maximum amplitude, initial phase and the waveform. A bowed string physical model might in turn have descriptors for fundamental frequency (or delay line length), bow pressure, bow position and bow velocity.

The low level descriptors can be either mappings from the higher level descriptors to the synthesis parameter space (e.g., pitch-to-frequency), or attributes that are exclusively associated with the synthesis techniques. In both cases, the low level descriptors are uniquely addressable entities. Two example descriptors are given in the table below. As other examples, a physical model of a violin would be controlled with a message '/mob/violin/bow/pressure 0.5', while a commercial synthesizer might have an addressable parameter at '/dx7/op1/eg/R1'.

Feature	Proposed OSC messages	Message description
low level descriptors	/frequency %f	continuous frequency in Hertz.
	/amplitude %a	maximum amplitude between [0..1]

### 4.3.5 MIDI over OSC

The following table proposes */midi* namespace, which wraps the MIDI 1.0 specification [4] as an OSC content format. The MIDI status codes are expressed as OSC address strings, while MIDI data bytes are encoded as OSC message arguments. MIDI channel is expressed explicitly by the instrument prefix described in 4.3.1, and thus not included in the messages below. The arguments are of OSC type *i* (int32), and their semantics are defined in [4]. 14-bit quantities are represented by a single OSC argument.

Feature	Proposed OSC messages	Message description
raw MIDI data	/midi %msg %s %d %d	0x00 status data1 data2 (packed into i32) separate arguments
MIDI channel messages	/midi/note/off /midi/note/on /midi/note/pressure /midi/controller /midi/program /midi/pressure /midi/pitchbend	%k %v %k %v %k %a %c %d %p [%b] %a %pb all arguments are integers <sup>1</sup> %k = key number, %v = velocity %a = pressure %c = controller number, %d = value %p = patch number, %b = bank number %a = pressure %pb = bend
MIDI system messages	/midi/sysex /midi/timecode /midi/songpos /midi/song /midi/tune /midi/clock /midi/start /midi/continue /midi/stop /midi/active /midi/reset	blob <sup>3</sup> %qf %spp %s - - - - - - - - all arguments are integers <sup>2</sup> %qf = quarter frame %spp = song position pointer %s = song number

<sup>1</sup> *k,a,c,p* are within [0...127]. *v,d,b,pb* are within [0...16383].

<sup>2</sup> *qf,s* are within [0...127]. *spp* is within [0.. 16383].

<sup>3</sup> bytes are within [0...127], bracketed with 0xF0 and 0xF7 pair

## 5 REFERENCES

- [1] Todorov, E. and Jordan, M. I. (1998). Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology*, 80(2):696–714.
- [2] Kilian J., 2001. Simple Image Analysis By Moments. Open Computer Vision (OpenCV) Library documentation.
- [3] Kim H., Moreau N., Sikora T., *MPEG-7 audio and beyond*, Wiley, 2006.
- [4] MIDI 1.0 Detailed Specification (with Addenda), Midi Manufacturers Association (MMA), <http://www.midi.org/techspecs/midispec.php>
- [5] Alexander R. Jensenius , Tellef Kvifte , Rolf Inge Godøy, Towards a gesture description interchange format, Proceedings of the 2006 conference on New interfaces for musical expression, June 04-08, 2006, Paris, France