Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders

D202.1 Report on repository standards, common interfaces and APIs (1st iteration)

Project Acronym	Prosperity4All
Grant Agreement number	FP7-610510
Deliverable number	D202.1
Work package number	WP201
Accessibility, Standards, Developers, Components, Programming Interfaces	Building Blocks
Authors	Till Riedel, Gregg Vanderheiden, Martin Deinhofer, Lukas Smirek, Simon Bates, Matthias Berning, Chris Veigl
Status	Final
Dissemination Level	Public
Delivery Date	16.03.2015
Number of Pages	37

Keyword List

Accessibility, Standards, Developers, Components, Programming Interfaces

Version History

Revision	Date	Author	Organisation	Description
11891	04/12/2014	Till Riedel	KIT	Initial Outline
12054	15/12/2014	Gregg Vanderheiden	RTFI	Edited Strategy
12154	18/12/2014	Martin Deinhofer	FHTW	Links added
12321	21/01/2014	Lukas Smirek	HDM	Input on various Standards
12522	29/01/2014	Simon Bates	IDRC	Input on Meteor
12577	02/02/2014	Matthias Berning	KIT	Input on Mobile Accessibility
12583	02/02/2014	Chris Veigl	FHTW	Input on Bio Signals
12584	02/02/2014	Till Riedel	KIT	Complete Draft Version
12864	02/02/2014	Till Riedel	KIT	Review Marco Aimar
13084	02/02/2014	Gottfried Zimmermann	HDM	Review

Table of Contents

List of Tables

1 Executive Summary

One year has passed in which components have been isolated, documented and gathered within the GPII Developer Space Component Listing. One important step was to identify common interfaces that not only will be usable by the current components but also to make components usable in the future.

This document documents a snapshot of the ongoing discussion to adopt common standards (i.e. specifications and interfaces). It defines the purpose of adopting and referring to standards within the GPII Developer Space Component Listing. We focus on existing standards, that are used by components or will guide the continuing convergence of components within the Developer Space. The current list and categorizes standards and interfaces within different relevant domains that are highly relevant for our progress.

NOTE: It is very important to note that the use of any standards is in no means mandatory for the inclusion of content to the Developer Space. This list is purely informative only gives a first guidance on what standards to consider and when to use them.

2 Standards and common APIs as prosperity strategy

2.1 Introduction and Methodology

The GPII uses standards wherever possible and appropriate. Because the GPII is an open project, use of standards facilitates its function both by making it easier for others to contribute to the effort, and by making it easier for other technologies to work with the GPII. The GPII both uses established public standards and creates its own internal standards for these purposes. The team working on the GPII also contributes to multiple international standards efforts that relate to its work (this work is addressed by the separate Standardization and Concertation task (T503) – a pertaining report P4A/D503.4 will be released in July 2015).

2.2 Purpose and structure of this document

The purpose of this document is to collect and document the different standards being used in the Developer Space that host potentially very heterogeneous technologies.

The first part (section 1) of the document explains the role of standardization and particular types of standards for the Developer Space.

The following parts (section 2-4) list different areas of standardization that are currently important within the Developer Space. This section has a reference character, but also should serve as basis for discussion on the scope of use of the mentioned standards.

Both parts are living documents that will be maintained throughout the project and are viewable and editable via http://wiki.gpii.net/w/Developer_Space/Standards on the GPII Wiki.

The view on concrete standards might change drastically considering the upcoming developments within the GPII and will be shaped by different interactions (such by the Prosperity4All Implementations). In 2015 a second edited revision of this document will be released that will reflect those developments.

2.3 Purpose of the Developer Space

With the Developer Space we are trying to collect and create components, frameworks, and tools that can make it easier to create new accessibility solutions both as assistive technologies, and as features to mainstream products.

The Prosperity4All Developer Space will also provide community resources and documentation that will help foster the adoption of accessible and flexible components.

(see http://wiki.gpii.net/w/Developer_Space)

2.3.1 Current state of the Developer Space

Currently the Developer Space is boot-strapped by creating a semi-structured listing of components (http://wiki.gpii.net/w/Developer_Space/Components) and resources (http://wiki.gpii.net/w/Developer_Space/Resources) on the common wiki with upcoming search functionality. In addition, discussions between developers and potential implementers have started via the mailing list (http://lists.gpii.net/cgi-bin/mailman/listinfo/dspace). Rather than being a fixed entity, the Developer Space is an evolving structure within the GPII that brings all kinds of component developers and product implementers together, which are interested in assistive technology and inclusive design.

As a commitment to Prosperity4All, developer teams have started integrating components with each other (such as the URC UCH middleware with spatial selection based on sensors or the Cloud4All matchmakers with AsTeRICS models). By looking at the needs of implementers, the goal is to come up with common APIs that reduce the cost of implementing multiple components and meeting the needs of different user groups.

2.3.2 Use of Standards and common APIs within the Developer Space

Not all of the components that will be in the Developer Space will have standard interfaces. Some of them are just code snippets or technologies developed by others that we have brought into the Developer Space.

Standards in this context can be existing standards or well-structured and well-documented common APIs (that can also be considered standards, see below). This convergence through standards and APIs is an important step towards the objective of cutting down the cost of improving the inclusiveness of IT and adapting new technologies for AT.

2.4 Possible reasons to not to standardize existing components

Standards typically comprise formal documentation for common (technical) interfaces, requirements and practices. Often, however, particularly in the internet domain, existing products or conventions between multiple parties become so accepted that they can be considered “de facto standards” even without the necessary formality. The reasons for using standards are many and well-documented (e.g. see http://www.iso.org/iso/home/standards/benefitsofstandards/ , http://archive.webstandards.org/edu_faq.html or http://wayback.archive.org/web/20081207115434/http://www.sutor.com/newsite/essays/e-OsVsOss.php). Particularly accessibility is an important aspect of many standards found in the web. Standards are nowadays essential parts of many ecosystems where many stakeholders are involved. Nonetheless we want to note potential reasons, why we may not

use “real” standards for things within the Developer Space or in special cases even not adapt to what is considered “de facto standard”.

- **“No existing standard is a good match for what is being done”**

A developer may not want to make the effort to adapt his work to something, that reduces effectiveness or functionality by forcing a fit to an existing standard meant for something different or more limited.

- **“Standardized niche products are still niche products”**

Niche products that need a lot of other adaption anyway (not out of the box usable) won't profit from standards. Other strategies, such as opening and modularizing the code might be better strategies of technical adoption.

- **“Standardization does not increase the reach and value”**

The applicable standards are not used widely enough in any target market to make the standardization effort worthwhile. On the other hand, standards might force you to make decisions that do not allow you to support or focus on a niche group. The actual reach and value of a component might actually not increase if it is standardized, but will become “yet-another” piece of software that attaches to a bigger effort.

- **“Standardization and markets are not consolidated enough”**

APIs are trending and fast moving, so are new standardization areas driven by multiple stakeholders and standardization organizations. If committing to one standard binds a lot of resources, it might be better to wait to see which community or vendor(s) take up the product, and identify which standards they would like to use, before investing in standardizing component interfaces and focus on functionality.

- **“Providing standardized interfaces on component level is not a value in itself”**

Ripping components out of an existing implementation in order to make them reusable is not a value in itself if there is no identified general use (or standard) for the component. Here, also other strategies like documenting may be of more benefit to those that would like to incorporate a component in their product.

Before considering or promoting standards these risks should be analyzed, documented and addressed. For this it is critical to reflect how much the use of a standard can contribute towards a goal, like wider adoption and sustainable development.

As the GPII supports the use and distribution of Open Source software it naturally also strongly encourages the use of Open Standards. Standards should most certainly not be used if they limit, rather than encourage, the distribution of components. Openness is an essential aspect of the Developer Space and should be supported by appropriate standards.

In the scope of different standardization organizations (such as the ITU-T or the IETF) or different national governments (such as the Danish "Definitions of Open Standards", 2004) or even laws (the Spanish "Ley 11/2007" of Public Electronic Access of the Citizens to the Public Services, 2007) the definition has been different. Today no general definition is established within the European Union while it at least gives a definition in the scope of eGovernment Interoperability (European Interoperability Framework for pan-European eGovernment Services, Version 1.0 (2004) ISBN 92-894-8389-X):

- ''The standard is adopted and will be maintained by a not-for-profit organisation, and its ongoing development occurs on the basis of an open decision-making procedure available to all interested parties (consensus or majority decision etc.). ''

- *The standard has been published and the standard specification document is available either freely or at a nominal charge. It must be permissible to all to copy, distribute and use it for no fee or at a nominal fee.*
- *The intellectual property - i.e. patents possibly present - of (parts of) the standard is made irrevocably available on a royalty-free basis.*
- *There are no constraints on the re-use of the standard.*

Particularly the last two bullet points differ from many other definitions that focus mostly on transparency. Similar definitions have been published by others, such as the Open Source Initiative (http://opensource.org/osr). Further reading may include Ken Krechmers "The Meaning of Open Standards" (http://www.csrstds.com/openstds.html).

2.5 Types of Standards and common APIs to consider

2.5.1 Guidelines and Requirement Catalogs

Many standards in accessibility are guidelines to support developers ensuring standardized definitions of accessibility. Governments and legislature often specify broad functional accessibility requirement that particularly apply for public services to set minimum standards that are particularly relevant for end products. Detailed requirement sets might be standardized, as in ETSI EN 301 549 "Accessibility requirements suitable for public procurement of ICT products and services in Europe". As with other guidelines, suppliers or producers of content or technical solutions may declare conformity with a complete standard or parts of it. In contrast to the above mentioned standards, those do not deal with technical interoperability. Particularly mainstream implementers will look for certain components to ensure conformance with functional accessibility requirements. An important question in the scope of the Developer Space is if functional requirements can be addressed or guaranteed on component level at all.

2.5.2 Technical Interoperability Standards

A particular subdomain is AT interoperability standardization that should support meeting accessibility guidelines within a certain IT domain. Those standards rather define interfaces than functional requirements. Particularly for systems that - unlike the Web - do not allow dynamic rendering of user interfaces, need specialized interfaces for accessing lower level functionality via accessibility APIs. As one example, ISO/IEC 13066-1:2011 is a standard for designing and evaluating interoperability between IT and AT.

It describes requirements for IT and AT products in three types of interoperability: hardware-to-hardware, hardware-to-software, and software-to-software. It also identifies a variety of software-to-software APIs that are described further in other parts of ISO/IEC 13066. These APIs can be used as frameworks to support IT-AT interoperability. IAccessible2, an extension of the Microsoft Active Accessibility API designed by IBM and now an open standard, is one of the APIs described in ISO/IEC 13066. (http://www-03.ibm.com/able/open_computing/iso.html)

2.5.3 De facto Standards and APIs

As seen by the aforementioned examples, many interoperability standards were originally driven by platform vendors as part of their operating systems, such as the Microsoft Active Accessibility, and UI Automation Specification, adopted by Microsoft products, the Apple Accessibility Toolkit or the Assistive Technology Service Provider Interface, mostly used by open source operating systems and particularly free desktops. Different APIs within the concrete GUI toolkits implement them either implicitly or with the need of explicit developer support (adding extra information). The IAccessible2 Standard, curated by the Linux foundation and based on Microsoft's Active Accessibility, is an example that shows the importance of market penetration for the success of standards.

2.5.3.1 Operating Systems

Desktop environments expose many interfaces that are relevant in our everyday work and personal life. Since the advent of the personal computer usability of those systems is an important factor for a large part of the population. According to internet user statistics by www.netmarketshare.com Microsoft Windows (91.45%) dominates the market with Apple MacOS (7.21%) and different flavors of Linux (1.34%) splitting up the rest of the market. The mobile operating system market in contrast is, according to the same source, split equally between Android (45.86%), developed mostly by Google on the basis of a Linux Kernel and iOS (43.15%), leaving Windows Phone only (2.28%) behind Symbian (4.62%) and JAVA ME implementations (2.92%).

Supporting particularly one of the major operating systems and their APIs can be considered a de facto standard. On the other hand, standards that are not well established within the ecosystem of one of those operating systems will have difficulties. While aforementioned efforts within ISO/IEC 13066 to standardize and converge AT interoperability has progressed to some degree, the accessibility APIs for mobile devices are still driven mostly by the vendors. While for desktop operating systems AT interoperability between operating systems is far from perfect, despite standardization efforts, more and more accessibility features are included in mobile operating systems. This is particularly interesting since the graphical user interface components are far more consistent on mobile devices so that once accessibility interfaces are implemented they may work for a wider share of applications (implicitly).

2.5.3.2 Frameworks, Architectures and APIs

Particularly in web-based applications underlying platforms play a much smaller role compared to desktop applications (for both server and client). Presentation and communication interface is widely standardized by Web Standards (see below). While standardized browsers and web views are available for most platforms, diverse web

frameworks that are used for content creation and rich web applications have become de facto standards beyond formal standardization. Often frameworks today are compiled of a small core and so-called plugins, modules and widget that are often developed by a wide range of third parties. The framework core can be seen as standard for the whole ecosystem. Examples are jQuery or angular.js. This is different from classical cross-platform GUI-Toolkits like QT or GTK+, that have a different type of development community.

One interesting development are cross-platform layers for web-applications like apache cordova or so-called polyfills (http://en.wikipedia.org/wiki/Polyfill), which try to support functionality before formal standard adoption. Both make particular use of HTML5 and JavaScript technology to provide a more uniform API landscape even before final standardization.

The role of the JavaScript as "the assembler language of the web" is further manifested in many server side APIs built in javascript via the node.js framework. Component frameworks developed within this community are often far less standardized than for example the dynamic component system defined by the OSGi Alliance for Java. Efficient execution of scripting languages together with platform as services providers, that run the code on web servers, has led to diverse ecosystems "in the Cloud". Particularly the lifecycle management for many applications has been shaped by cloud computing on one hand and mobile apps on the other.

Today this often means much tighter interactions between components than foreseen by many of the standards that were driven by service oriented computing paradigms. Nonetheless service interfaces e.g. defined by the OASIS play an important role particularly in different business domains.

"The Hypermedia as the Engine of Application State" proposed by Roy Fielding is becoming more a reality than fixed, standardized interfaces. Today many interfaces reference to REST as a standard: GET, POST, PUT and DELETE in combination on resources identified by uniform resource locators (often in combination with JavaScript Object Notation or eXtensible Markup Language payloads) often serve as a common denominator in many applications today (not only in the Web). More and more formal standards define functionality on those minimal interactions. Particularly in new frontiers for IT systems like Smart Homes or the Internet of Things, REST Services together with other Web Technology are picking up. Looking at the fact that accessibility played an important role in the World Wide Web from the beginning this can be seen as an important opportunity to provide accessible interfaces also in a "Web of Things".

2.5.4 GPII supported standards and APIs

Within the GPII there are many ongoing standardization activities. Following a "eating your own dog food" approach, those standards should be particularly promoted within the GPII.

However, also in terms of enabling a maximum interoperability with Developer Space/Components with the infrastructure parts of the GPII, adopting those standards should be a priority.

More of those activities will be described as a result of the cross-cutting activities within the P4A project. All work on standardization is and will be gathered on the wiki.

In addition, the GPII is adopting common Technical Standards that should guide the work in the Developer Space.

3 Categorizing Functionality, Requirements and Preferences

An important factor for the Developer Space will be that users (i.e. implementers) find useful tools and that developers can offer their tools in a fashion that they are found specifically by stakeholders in the accessibility domain. Therefore it is wise to look particularly at categorization schemes. While this might be not so important for mainstream software developers looking for concrete functionality, many non-classical developer types we want to address, like AT product developers, healthcare professionals or relatives have a domain specific view. Furthermore procurement might be a driving factor for accessibility improvements and the need for adoption of components, so that alignment with existing schemes might prove as an opportunity in some cases.

3.1.1 Assistive products for persons with disability -- Classification and terminology (ISO 9999:2011), Global Medical Device Nomenclature (GMDN) and Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT)

The Global Medical Device Nomenclature (GMDN) is a comprehensive system of internationally recognized coded descriptors in the format of preferred terms with definitions used to generically identify and characterize types of medical devices.

ISO 9999 is another standard particularly focused on assistive products. It offers both classification and terminology for specific AT product functionality and also classification means for adaptive technologies and products. The standard is maintained by the ISO and is continuously updated to meet technological developments and new needs. Every assistive technology is assigned a six digit code within a three stage product hierarchy.

The following table shows the top level classes.

Table 1: Assistive product hierarchy according to ISO 9999

top class	name	defined subclasses
Class 04	Assistive products for personal medical treatment	15 subclasses
Class 05	Assistive products for training skills	11 subclasses
Class 06	Orthoses and prostheses	12 subclasses
Class 09	Assistive products for personal care and protection	19 subclasses
Class 12	Assistive products for personal mobility	14 subclasses

top class	name	defined subclasses
Class 15	Assistive products for housekeeping	5 subclasses
Class 18	Furnishings and adaptations to homes and other premises	11 subclasses
Class 22	Assistive products for communication and information	13 subclasses
Class 24	Assistive products for handling objects and devices	13 subclasses
Class 27	Assistive products for environmental improvement, tools and machines	5 subclasses
Class 30	Assistive products for recreation	11 subclasses

The WHO has made efforts to map those classes to diseases and functioning profiles (http://whqlibdoc.who.int/hq/2010/WHO_HSS_EHT_DIM_10.2_eng.pdf) to identify Priority Medical Devices (PMD) based measuring the burdens of diseases world wide. Also, the International Health Terminology Standards Development Organization (IHTSDO) has set out to improve health of humankind by providing standardized clinical terminologies. The Systematized Nomenclature of Medicine – Clinical Terms (SNOMED-CT) even provides formal, logic-based definitions based on a multilingual thesaurus to be processed electronically. SNOMED-CT provides an overlapping hierarchical scheme and triple relations that enable description logic based reasoning. It also provides a subdivision of assistive products that however, does not directly map ISO 9999. More detail, particularly on ISO 9999, is given in the International Encyclopedia of Rehabilitation.

3.1.2 User Profile Management (ETSI EG 202 325) and User Profile Preferences and Information (ETSI ES 202 746)

A set of ETSI standards specifically aims at user profile providers, operators, service developers, service providers, device manufacturers and standards developers. The ETSI EG 202 325 standard describes how profile information should be managed. “Special needs” regarding accessibility are seen as part of such preferences. The ETSI ES 202 746 standard specifies information and preferences, which are choices made by the user, that will result in driving the behavior of the system, and builds on the user profile concept described in ETSI EG 202 325. Profile solutions following the standard should be provided for the primary benefit of the end-user and give end-user the rights to manage the profile contents. It is interesting because it addresses auto-personalization but acknowledges particularly the right for informational self-determination.

3.1.3 Accessibility guidelines for information/communication technology equipment and services (ISO 9241-20)

ISO 9241, which addresses "Ergonomic requirements for office work with visual display terminals", contains in its part 20 a set of guidelines that can help to ensure that ICT equipment can be used by the widest range of people, regardless of their capabilities or disabilities, limitations or culture. It follows a very holistic understanding of inclusive IT that covers disabilities from birth as well as for elderly people, temporary or situational disabilities, and aims at guaranteeing ergonomic use regardless of physical, sensory and/or cognitive impairments.

3.1.4 Information technology — Individualized adaptability and accessibility in e-learning, education and training (ISO/IEC 24751)

This standard applies to personalization on demand enabled by ICT and networked systems distinguishing:

- Display: how resources are to be presented and structured;
- Control: how resources are to be controlled and operated; and,
- Content: what supplementary or alternative resources are to be supplied.

Similar to ISO 9241, it is not restricted to classic notions of disability and takes an integrative approach. It is intended for all users as every user can experience a mismatch of their individual needs and preferences and the content or services delivered. While the first part defines a framework, particularly the second part is interesting in the scope of categorization. This categorization focuses on the functional abilities and the assistive technology or other non-standard technology in use and distinguishes itself clearly from medical approaches and standards that focus on naming and describing human impairment. The standard provides attribute descriptions and usage recommendations for common adaptation preferences.

ISO 9241 is currently being revised with input from GPII. The common terms registry is a major contribution of the GPII to that effort that implements a software component for managing common terms. The required fields and procedures entry and maintenance of the Common Terms Registry are being defined by the new ISO 24751. The needs and preference set used for auto-personalization build upon those common terms to let the user specify what they want/need/prefer the environment to look or behave like. Consequently adopting common terms will help components of the Developer Space particularly interact with the infrastructure components provided by the GPII. Either adopting or mapping of component preferences to terms will enable auto-personalization based on the existing matchmaking features. Further information on the standardization can be particularly found on the AccessForAll Working Group wiki.

3.1.5 IMS Global Access for All Information Model Data Element Specification and Accessibility Web Schemas

The IMS Global Learning Consortium specifies a formal information model that adapts Access for All specification. IMS has licensed these properties to schema.org under the Creative Commons Attribute-ShareAlike License so that they can be used under schema.org's corresponding terms and conditions. The a11ymetadata website links applications of the specification and its relationships with previous metadata efforts.

The interesting part of the effort to align with schema.org, a general metadata tagging effort launched by major search engine operators, is that accessibility requirements can be used as search filters by custom web searches. Examples include particularly filtering online videos by captioning. Early adoptions demonstrate how to describe resources particularly, how meta-data can be included into learning resources in different scopes. An example that is particularly relevant for the media transformation components in the Developer Space is tagging of accessible media in terms of accessibility features, hazards and control functionality as visible by using the google richsnippets viewer on selected examples from the accessible online library bookshare.

Schema.org primarily uses the WHATWG MicroData specification to embed ontologies directly into web content. The vocabulary builds primarily on existing standards and best practices. In contrast to other efforts, it has been supporting multiple syntaxes, specifically including RDFa and JSON-LD (http://www.schema.org/docs/faq.html). This particularly allows embedding content from very different rich web applications and mash-ups that may mix different vocabularies. More information can be found via the W3C page on Accessibility WebSchemas.

3.1.6 Accessibility requirements suitable for public procurement of ICT products and services in Europe (ETSI EN 301 549)

ETSI EN 301 549 is a catalog of "functional accessibility requirements applicable to ICT products and services". Among other aspects, it also elaborated related test descriptions and evaluation methodology to a level of detail compliant with ISO/IEC 17007:2009. It aims primarily at public procurers to standardize the requirements for their call for tenders and at manufacturers to address standardized accessibility requirements within their design, build and quality control. The EU mandated to European Standardization to actually standardize requirement lists for certain ICT products on the basis of this standard. One of the outcomes of this mandate is a generator for standardized call for tender texts (http://mandate376.standards.eu).

A generated tender for cloud computing applications will contain text like: The offered solution shall meet Clauses 5.2, 5.3, 5.4, 5.5.1, 5.5.2, 5.6.1, 5.6.2, 5.7, 5.8, 5.9, 6.2.1.1, 6.2.1.2, 6.2.2.1, 6.2.2.2, 6.2.3, 6.2.4, 6.3, 6.4, 6.5.2, 6.5.3, 6.5.4, 6.6, 7.1.1, 7.1.2, 7.1.3, 7.2.1, 7.2.2, 7.2.3, 7.3, 9.2.1, 9.2.2, 9.2.3, 9.2.4, 9.2.5, 9.2.6, 9.2.7,

9.2.8, 9.2.9, 9.2.10, 9.2.11, 9.2.12, 9.2.13, 9.2.14, 9.2.15, 9.2.16, 9.2.17, 9.2.18, 9.2.19, 9.2.20, 9.2.21, 9.2.22, 9.2.23, 9.2.24, 9.2.25, 9.2.26, 9.2.27, 9.2.28, 9.2.29, 9.2.30, 9.2.31, 9.2.32, 9.2.33, 9.2.34, 9.2.35, 9.2.36, 9.2.37, 9.2.38, 9.3, 10.2, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, 10.2.6, 10.2.7, 10.2.8, 10.2.9, 10.2.10, 10.2.11, 10.2.12, 10.2.13, 10.2.14, 10.2.15, 10.2.16, 10.2.17, 10.2.18, 10.2.19, 10.2.20, 10.2.21, 10.2.22, 10.2.23, 10.2.24, 10.2.25, 10.2.26, 10.2.27, 10.2.28, 10.2.29, 10.2.30, 10.2.31, 10.2.32, 10.2.33, 10.2.34, 10.2.35, 10.2.36, 10.2.37, 10.2.38, 10.2.39, 10.2.40, 11.2, 11.2.1.1, 11.2.1.2, 11.2.1.3, 11.2.1.4, 11.2.1.5, 11.2.1.6, 11.2.1.7, 11.2.1.8, 11.2.1.9, 11.2.1.10, 11.2.1.11, 11.2.1.12, 11.2.1.13, 11.2.1.14, 11.2.1.15, 11.2.1.16, 11.2.1.17, 11.2.1.18, 11.2.1.19, 11.2.1.22, 11.2.1.23, 11.2.1.24, 11.2.1.25, 11.2.1.26, 11.2.1.27, 11.2.1.28, 11.2.1.29, 11.2.1.30, 11.2.1.31, 11.2.1.32, 11.2.1.33, 11.2.1.34, 11.2.1.35, 11.2.1.36, 11.2.1.37, 11.2.1.38, 11.3.2.1, 11.3.2.2, 11.3.2.3, 11.3.2.5, 11.3.2.6, 11.3.2.7, 11.3.2.8, 11.3.2.9, 11.3.2.10, 11.3.2.11, 11.3.2.12, 11.3.2.13, 11.3.2.14, 11.3.2.15, 11.3.2.16, 11.3.2.17, 11.4.1, 11.4.2, 11.5, 11.6.1, 11.6.2, 11.6.3, 11.6.4, 11.6.5, 12.1.1, 12.1.2, 12.2.2, 12.2.3, 12.2.4, 13.2, 13.3 from EN 301 549"Accessibility requirements suitable for public procurement of ICT products and services in Europe", available at http://www.etsi.org/deliver/etsi_en/301500_301599/301549/01.01.01_60/en_301549v010101p.pdf

The supplier may answer such requirement by third party certification or a declaration of conformity. On one hand the risk of such procurement policies is the risk of under-specification by not requiring supporting a whole standard. On the other hand it will lead to a quite long list of individual requirements like the above that makes it difficult for small vendors to answer such tenders. Nonetheless such practice is becoming common throughout Europe and it is viable to ask if such modular requirement lists can be referenced by modular components.

4 Platform Level Accessibility

4.1 Interoperability between Information Technology (IT) and Assistive Technology (AT) (ISO/IEC 13066)

While accessibility has become an essential part in many operating systems and is mainly driven by or through the vendors, ISO/IEC 13066 has set out to define interoperability with Assistive Technology on a platform level. The first part 13066-1 is a cross-platform standard for designing and evaluating interoperability between IT and AT regarding three types of interoperability: hardware-to-hardware, hardware-to-software, and software-to-software.

Part 2 of this standard defines, among other things, the Windows Accessibility Application Programming Interface (API) that was de facto standardized before via the Microsoft Windows Automation Frameworks, including Microsoft Active Accessibility, User Interface (UI) Automation, and the common interfaces of these accessibility frameworks including the IAccessibleEx interface specification. This standardization aims only at one operating system. Still, it provides stability particularly towards AT Vendors as it formally specifies services provided in the Microsoft Windows platform to enable AT to interact with other software within Windows. Products are most reliable when standardized public interfaces are used. It describes requirements for IT and AT products in three types of interoperability: hardware-to-hardware, hardware-to-software, and software-to-software. It also identifies a variety of software-to-software APIs that are described further in other parts of ISO/IEC 13066. These APIs can be used as frameworks to support IT-AT interoperability.

In its third part, the standard defines IAccessible2, an extension of the Microsoft Active Accessibility API designed by IBM and now an open standard. IAccessible2 constitutes a cross-platform approach to desktop accessibility, as an extension of the Microsoft Active Accessibility API designed by IBM. It bridges the gap towards Accessible2 fills in perceived omissions in MSAA to match the Java Accessibility API and Assistive Technology Service Provider Interface (AT-SPI) commonly used on Linux Desktop Systems.

4.2 Accessibility in Mobile Platforms

As mentioned above, all major smartphone platforms provide support and guidelines for development of accessible technology. It should be noted that these are only de facto standards and they are only encouraged, but not mandatory for publication of applications. Even the strict reviews of Apples AppStore do not check for accessibility. All platforms have checklists as well as testing tools and frameworks to design for accessibility.

Although having only a minimal market share, Microsoft already has extensive support for AT in their mobile operating system and guides developers how to design accessible applications.

This stems from the fact that they encourage the development of unified applications, which run on mobile and desktop devices alike. Therefore, they are building on their extensive experience described above. Apple also transferred their knowledge from the desktop platform, although different mechanisms and guidelines are imposed for both systems. The main focus of the iOS guidelines, similiar to the other manufacturers, is the compatibility with the text-to-speech function, which is provided system-wide. API functions are therefore focused on providing captions and labels, but there are also recent additions to specify custom actions for accessible interfaces. Android is very open for enhancements and provides the possibility to register background services that handle different input and output-modalities. Therefore a user is able to customize his interface through installation of matching services via the app store, given that the used applications include the necessary annotations.

A special case are cross-platform applications, which are built with Web technologies and are executed in a browser view on each of the operating systems. These are usually created with frameworks like Apache Cordova, which generated the application packages for the different platforms. On one hand, they already support the accessibility technologies of the browser view, used for rendering and profit from the efforts made for Web technologies. On the other hand, they are detached from the possibilities provided by the base operating system and do not benefit directly from neither the optimizations of native UI elements, nor tools and mechanisms provided by the manufacturer. Therefore, additional plugins are needed to access the platform specific APIs.

5 Web Accessibility

Given the GPIIs focus on Internet technology, a main focus of standardization is Internet technology. Particularly standardization regarding Application, Presentation and Session protocols (OSI layer 5-7) are important in the scope of the GPII. In the internet this technology is summarized by the World Wide Web, and mostly standardized by the W3C (in contrast to the IETF for lower networking protocols). Particularly APIs supported by Web Browsers are standardized by the W3C. The most notable standards are HTML, XML, CSS and DOM.

However, the most common scripting language is standardized by the ECMA (ECMAScript aka JavaScript). These standards, however, are typically not competing but referencing each other. JavaScript was established as a de facto standard by Netscape and contributed to the ECMA for standardization under ECMA-262. JavaScript is a good example of the varying scope of standardization (ECMA does not standardize all extensions that are commonly supported by JavaScript). It is also an example where standardization has led to competing standards, i.e. JavaScript Object Notation (JSON, ECMA-404, RFC 7159) that is commonly used in many components of the GPII. It is good for developers to be aware of this. However, practical cross-browser support (de facto standardization) and actively tested interoperability is far more relevant.

The most important standard that is shaping the Web is HTML (that primarily describes the structuring and presentation of content for web browser). An important development with the final standardization of HTML5 in 2014 is that many technologies that were part of the original HTML5 standardization process were moved to separate specifications (often maintained outside the W3C):

- HTML Canvas 2D Context
- Web Messaging,
- Web Workers,
- Web Storage,
- WebSocket API,
- Server-Sent Events,
- WebSocket Protocol,
- WebRTC
- WebVTT

Other extension that are referenced by the HTML standard most notably include:

- SVG
- MathML
- WAI-ARIA

All those web standards are important for the Developer Space as a common denominator. These extensions particularly offer many innovative uses of Web Technology for assistive

technology. It is, however, clear that even this set of standards is very wide and only defines APIs and markup that can be used by web application that can run across multiple vendors. Particularly those specifications do not guarantee interoperability of different components and compatibility with commonly used assistive technology out of the box. Lately the scope of HTML5 even broadened as one of the most promising frameworks for cross-platform for mobile applications.

As Web standards have been designed with accessibility in mind, the extension of the scope of Web standards towards mobile application is a huge opportunity for tackling accessibility barriers in many domains beyond the stationary Internet. Thus a focus of this document is dedicated towards relevant web Standards as a potential for future assistive technology and inclusive component design.

5.1.1 Web Content Accessibility Guidelines (WCAG)

One of the major milestones in IT Accessibility were the Web Content Accessibility Guidelines. The first version was published by Gregg Vanderheiden just after accessibility was raised in the second World-Wide Web by Tim Berners-Lee.

The current version 2 of Web Content Accessibility Guidelines aim at making Web content more accessible for people with all kinds of disabilities, but also for elderly people. The goal of WCAG 2.0 is providing a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally.

WCAG is primarily intended for:

- Web content developers (page authors, site designers, etc.)
- Web authoring tool developers
- Web accessibility evaluation tool developers
- Others who want or need a standard for web accessibility

Thus, the WCAG as a fundamental guideline plays an important role particularly for the web components and media transformation components, but is also applicable throughout the Developer Space (particularly the very own interface of the Developer Space listing).

At the top are four principles that provide the foundation for Web accessibility: perceivable, operable, understandable, and robust.

Under the principles are guidelines. 12 guidelines provide the basic goals that authors should work toward in order to make content more accessible to users with different disabilities. For each guideline, testable success criteria are provided to allow WCAG 2.0 to be used where requirements and conformance testing are necessary such as in design specifications, purchasing, regulation, and contractual agreements. According to those guidelines three levels (A, AA, AAA) of accessibility conformance can be reached.

The standard is explained for adopters in multiple supporting Documents:

1. How to Meet WCAG 2.0 - A customizable quick reference to WCAG 2.0 that includes all of the guidelines, success criteria, and techniques for authors to use as they are developing and evaluating Web content.

2. Understanding WCAG 2.0- A guide to understanding and implementing WCAG 2.0. There is a short "understanding" document for each guideline and success criterion in WCAG 2.0 as well as key topics.

3. Techniques for WCAG 2.0- A collection of techniques and common failures, each in a separate document that includes a description, examples, code and tests.

(Source: http://www.w3.org/WAI/intro/wcag.php)

There is a diagram and description of how the technical documents are related and linked. Further educational materials available via the working group homepage within the W3C.

5.1.2 Accessible Rich Internet Applications Suite (WAI-ARIA)

While HTML generally should be accessible, WAI-ARIA helps with rich internet applications using dynamic content and advanced user interface controls. It is currently ranked as an official extension to HTML5. Especially it focusses on providing web developers the ability to better support people who rely on screen readers and people who cannot use a mouse. Particularly it does so by defining:

- Roles
  - to describe the type of widget presented, such as "menu", "treeitem", "slider", and "progressmeter"
  - to describe the structure of the Web page, such as headings, regions, and tables
- Properties
  - to describe the state that widgets like a check box are in
  - to define dynamic regions of a page that are likely to get updates
  - drag-and-drop that describe drag sources and drop targets
- A way to provide keyboard navigation for the Web objects and events, such as those mentioned above.

Commonly a lot of HTML5 based user interface frameworks use WAI-ARIA as an interface for providing advanced accessibility. Often this is still done via special templates or plugins or special configuration is needed. The Web Accessibility Component section links a few of those plugins or guides. The Mozilla Developer Networks also give a lot of useful links regarding WAI-ARIA support.

6 Realtime Messaging & Eventing

Realtime Messaging and Eventing are important to connect components in the Developer Space. Particularly human input devices and multimodal output need to connect in a reactive fashion. As the project focusses on adaptation of web standards wherever possible, a focus is on Web Technology.

6.1 Relevant General Standardization Activities

6.1.1 WebSockets

Recently low latency client server interaction has become an important focus of current Web Applications. This development started with AJAX (Asynchronous JavaScript and XML), which builds on the browser capability to asynchronously process structured payload received from a server. Nevertheless, such systems are bound to classical polling from the client. Efficient implementations are achieved by blocking calls at the server (coined “long polling”, “Push” or “Comet”). True bidirectional communication between server and client is not possible and timing and reactiveness is often an issue. Emulating true BSD Sockets (the technology used to interface network functionality in most operating systems) the WebSocket Standard (https://tools.ietf.org/html/rfc6455) defines a widely adopted extension to the HTML5 specification (http://caniuse.com/#feat=websockets). Beyond Webservers projects like http://autobahn.ws/ demonstrate the wide usability of this interface for multiple components on top of web technology even without classical web browsers.

One downside of WebSockets is that it does not build upon HTTP as ubiquitous transfer protocol within the web but rather gives a native interface to Web Browsers to TCP functionality, while only maintaining many basic HTTP protocol headers. It is unclear how far WebSockets themselves lead to higher interoperability between components as they specify only a transport layer. They do, however, enable TCP like communication with web browsers, which can be important for e.g. connecting frameworks like AsTeRICS to Web based infrastructures.

As a first contribution to the AsTeRICS project, the AsTeRICS runtime supports the WebSocket protocol as of version 2.5.

Advanced functionality is typically de facto standardize via libraries like socket.io or autobahn.ws, that also provide fall-back to polling options. One important possibility to extend this standard for bidirectional communication is the use of so-called Protocol Extension (http://chimera.labs.oreilly.com/books/1230000000545/ch17.html#_protocol_extensions). Those extensions can be negotiated between components in a standardized way. One

extensions for multiplexing connections is described in a HyBi Working Group draft by Google. Beyond wrapper APIs other open standards like the web application messaging protocol have been proposed that work on top of web sockets, that e.g. specify remote procedure calls or publish/subscribe interaction in open systems. Other solutions, like the Distributed Data Protocol specified by meteor www.meteor.com/ddp also build on top of WebSockets and allow to get data structure from a server and to automatically propagate changes throughout the network.

6.1.2 Server Sent Events

Server-Sent Events (SSE) are an alternative to WebSockets that build upon standard HTTP. In contrast to WebSockets it does not provide bidirectional communication, but provides a well integrated interface for asynchronous events from a server without the need for multiple connections. In JavaScript a standardized API called EventSource is implemented by nearly all current major browsers (http://caniuse.com/#feat=eventsource). A polyfill library is, however, available that also implements the functionality on older browsers, except for some android browsers and Opera Mini. A good introduction is available via HTML5Rocks that shows the advantages in terms of simplicity of this communication technology. In contrast to WebSockets, SSE specifies a clear publish/subscribe scheme that provides a tighter framework for interoperability, where WebSockets would require further specification.

6.1.3 WebRTC

Another extension to the HTML5 standards is WebRTC, which targets realtime messaging. WebSockets compare to WebRTC much like TCP to the Real Time Messaging Protocol (RTMP). It also integrates with current web browsers and particularly provides peer to peer communication. It is developed particularly with video-conferencing applications in mind and currently has less adopters in the browser landscape (http://caniuse.com/#feat=rtcpeerconnection). However, support is growing fast and the protocol provides radically new possibilities for interaction between clients. The protocol does not only allow realtime audio and video streaming but provides support for arbitrary datagrams (https://www.webrtc-experiment.com/DataChannel/). Currently studies are running if the crowd-sourced activity recognition systems in the Developer Space can be ported to this technology. The advantage of peer to peer communication will be much lower server costs for service providers as a server is needed only in the setup phase not during communication, also privacy issues can be much better handled with this type of technology. http://peerjs.com/ and http://rtc.io are two javascript APIs that wrap WebRTC functionality in a more usable way for general purposes.

6.1.4 Message Queue Telemetry Transport (MQTT)

While the above protocols that are HTML5 extensions are relatively easy to use within web browsers, particularly hardware provides have been looking at even more lightweight protocols to connect to the so-called “machine-to-machine” (M2M) or “Internet of Things” world of connected devices. MQTT is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. It was originally invented by Dr. Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom back in 1999 but became a de facto standard for many networked embedded applications. Since October 2014 MQTT has become an official OASIS Standard. MQTT has a broad community and its wiki page list a wide range of implementations for both clients and servers.

6.1.5 Dynamic Discovery

Another technology stack that builds upon HTTP based web services is the Devices Profile for Web Services (DPWS), that was meant to replace particularly the similar UPnP standard (which still has broad adoption in the personal multimedia domain). Unlike UPnP, which builds upon a device state notion, DPWS is fully compliant with existing WS-* Specifications and implements a service oriented architecture.

The advantage over above mentioned standards is that those protocols operate without any infrastructure using multicast discovery. Nonetheless interoperability of the advanced functionalities like WS-Eventing has been problematic in current implementations on various reports. Furthermore, although building on HTTP and XML, no direct support for web-browser is given (partially because HTTP over UDP is used for service discovery). Projects like WS4D provide DPWS stacks for a wide range of embedded devices and client support is e.g. included in the Microsoft Windows Communication Foundations.

Other standards are on the way to even allow physical discovery using web based mechanisms. Very recently Google started an initiative to use Bluetooth technology (see below) to identify physical objects called URIBeacons. The technology is similar to the use of QRCodes and other 2D Barcodes that have become a de facto standard for interlinking physical and Internet content. In contrast also to Near Field Communication (NFC), URIBeacons, or the proprietory Apple Product iBeacon, allow identifying objects in the proximity without the need to physically touch an object or visually identify an object. In contrast to network discovery like UPnP or DPWS it allows to limit found devices to the immediate physical proximity.

6.1.6 COAP Notifications

Particularly for resource efficient devices that want to connect to web based infrastructure the Constrained Application Protocol (CoAP) provides a binary HTTP alternative for M2M communication. Lately, hardware vendors have pushed this standard with the Internet Engineering Task Force (IETF) Constrained RESTful environments (CoRE) Working Group, that is specified in RFC 7252. Standards for notification and eventing using CoAP have been proposed e.g. in a draft specification. (see below)

6.1.7 Open Sound Control (OSC)

Due to the fact that standardized realtime messaging is commonly difficult to implement and that open standards and open implementations are rare, a lot of applications have been developed on top of the open sound control specification, even if no sound or music information is transported.

There are several relevant components for the Developer Space that already support the OSC Protocol, e.g. different brain computer interface projects like OpenVibe, which is also supported by the AsTeRICS runtime. AsTeRICS also supports the OSC protocol to connect further remote components.

6.2 Domain Specific Standards in P4A Building Blocks

Many software components exposed in the Developer Space not only should implement a domain independent messaging API towards a bigger software system, but also need to interface with very specific components, explicitly hardware. Those domain specific message interfaces are mostly component specific.

6.2.1 Autopersonalization

Enabling autopersonalization is the primary objective of the GPII Architecture. More information on standards can e.g. be found in this list of standards related to personalization.

6.2.2 Remote Control

The ISO/IEC 24752 standards specify the basic concepts and document formats for the Universal Remote Console ecosystem. In its second edition, the URC technology has been revised based on lessons learned to be simpler to implement and based on current technologies. The URC technology is part of the GPII concepts. Through its separation of frontend and backend, it allows for pluggable user interfaces that may be provided by the manufacturer of a device or by third parties. In the AAL context, URC is a basic technological

concept for auto-personalization by preferences that particularly targets control of devices and services.

URC particularly provides for abstracting different target protocols. The new part 6 of ISO/IEC 24752 defines a way for devices/services to expose themselves as SOAP-based Web services in a URC-compliant way. The Universal Control Hub can be construed as a proxy target that provides access to other targets. The URC-HTTP protocol of the UCH and the URC-HTTP Target protocol is predecessors for a new RESTful protocol for targets to be standardized (see the GPII Standardization Roadmap).

Device templates are strongly needed for interoperability of pluggable user interfaces across device models. Device manufacturers can use the appropriate templates as a basis for their products. With the new inheritance feature (see ISO/IEC 24752-2:2014) user interface sockets can be easily extended for product-specific additions. However, the pluggable user interfaces for the device template can still be used for the extended product (albeit hiding the additional features). A set of device templates consists of the following documents:

- Target description template
- Socket description template
- Grouping sheet template
- Resource sheet template
- Optional: WSDL1 and WSDL2 documents

One contribution of the Prosperity4All Developer Space are so called URC Super Sockets that will work for multiple targets and thus lower the barriers for adapting this technology for new devices.

6.2.3 Smart Home Integration Modules

Smart Homes are an important factor for real world accessibility beyond IT systems. Particularly this domain is driven by specific hardware vendors. While many proprietary interfaces and bus systems exist, different efforts have been on the way to standardize components.

The KNX bus system is used particularly in commercial buildings but also increasingly in home installations. KNX originated from a long run effort to standardize a “European Installation Bus”. It is currently the most widely adopted installation bus in professional installations (http://knx.org/knx-en/knx/technology/introduction/index.php).

Particularly the manufacturer EnOcean, who is providing patented energy harvesting solutions for switches, has pushed his wireless protocols through standardizations and has formed a broad alliance. EnOcean products are very popular for retrofitting Smart Homes. Both KNX and EnOcean are supported by the Smart_Home_Integration_Modules in the Developer Space.

Particularly in the wireless domain a huge diversity exists. Most those systems are proprietary and focus on specific simple to retrofit functionality like smart switches and plug (e.g. Plugwise), lighting (e.g. Philips Hue) or window blinds (e.g. Somfy) . However, equipment is typically more pricy than alternatives like the FS20 and FHT Systems that are particularly used by homeowners for retrofitting smart home functionality. The FS20 particularly supports a wide range of cheap wireless equipment that allows retrofitting.

A trend is opening particularly those systems by providing either WIFI-Gateways with REST-Interfaces or USB adapters with simple serial protocols. With the SmartM2M Standard that particularly also includes the KNX Systems a broader standard for a REST-based interface is on the way. However, at the moment also other vendors like the Google owned Nest are starting initiatives to establish an internet ecosystem that might become a de facto standard for many systems before open standardization can come up with solutions.

One important development is the broad community support for multi-protocol frameworks like FHEM or the Eclipse Smart Home (formerly OpenHAB) as well as multiprotocol HW Gateways like that today already allow programmatic interfacing across multiple proprietary and open protocols using a modular architecture. Other research driven frameworks like the Universaal Middleware also include multiple adaptors. Commonly such middleware is used to support novel ambient assistant living applications (see T.Zentek, et.al, Which AAL Middleware Matches My Requirements?, Springer 2015). Particularly the Universal Remote Console Standard provides a potential commercial adoption by abstracting and adapting different target devices uniformly. Current work is underway in P4A that connects Eclipse Smart Home functionality with the Universal Remote Console standard.

6.2.4 Bioelectrics Signal Acquisition and Processing Modules

Bioelectric Signal Acquisition and Processing Modules

Bioelectric signals are used in various applications, ranging from clinical screening and rehabilitation therapy to assistive technology and alternative Human Computer Interfaces (HCI) and Brain Computer Interfaces (BCI). A variety of vital parameters can be acquired and classified, including the Electroencephalogram (EEG), the Electrocorticogram (ECoG), the Electrooculogram (EOG), the Electromyogram (EMG) as well as the Elecrodermal activity (EDA).

For assistive technology applications and alternative HCI, usually non-invasive (surface mounted) biosignal acquisition techniques are in use. The bioelectric potentials are measured via electrodes (Ag/AgCl, gold-plated or skin pads) after proper skin preparation (using alcohol for skin cleaning an electrode gel).

The utilized technical equipment for the acquisition of bioelectric signals usually consists of a signal amplification circuit (comprising instrumentation amplifiers and analogue filter stages), proper digital-to-analog conversion (ADC, resolution ranging from 10 up to 24 bit and sampling rate ranging from 100 Hz to several kHz). Further filtering, feature extraction and classification is usually performed in the digital domain. For BCI applications, various paradigms exist which imply different signal processing and feature extraction methods, for example event-related desynchronisation (ERD), P300, or steady-state visual evoked potentials (SSVEP) (see http://arxiv.org/pdf/1309.2055.pdf)

In recent years, many SOC-manufacturers entered the market, so that nowadays integrated biosignal acquisition IC's with multichannel processing and 24 bit ADCs are available for a reasonable price (e.g. ADAS1000 by Analog Devcies http://www.analog.com/en/analog-to-digital-converters/ad-converters/adas1000/products/product.html or the ADS1299 by Texas Instruments, http://www.ti.com/product/ads1299). Due to this development, biosignal acquisition equipment became more affordable and even entered the mass market (including gaming applications - e.g. the Emotiv EPOC headset https://emotiv.com/epoc.php).

Two open source community driven projects which allow biosignal aquisition for assistive technology applications are covered by WP202 building blocks:

- The OpenEEG: http://openeeg.sourceforge.net/doc/modeeg/firmware/modeeg-p2.c
- The OpenBCI: https://github.com/OpenBCI/Docs

These data acquisition units include a microcontroller which emits a proprietary data format (called "P2" for the OpenEEG derivates and "OpenBCI Data format" for the OpenBCI). These data formats include only framing information and the binary channel data - no patient information etc. The bioelectric signal acquisition building blocks which are being developed in course of WP202 will make the channel data available to GPII / P4All developers and/or implementers via a defined API.

International standardization in the field of bioelectric signal acquisition and processing has been difficult and progressing slowly due to many different manufacturers, proprietary products and applications. Major standardization efforts include clinical data exchange and device interoperability. Notable bodies are the Continua Health Alliance (http://www.continuaalliance.org/) and HL-7 (http://www.hl7.org/).

The ISO/IEEE 11037 standard for point-of-care / health device communication enables communication between medical, health care and wellness devices and with external computer systems. Furthermore the Bluetooth / Bluetooth 4.0 wireless communication standard defines several Health Device Profiles (HDP) (see: https://developer.bluetooth.org/TechnologyOverview/Pages/HDP.aspx)

Despite all these efforts, no standardized data format could be established as a de-facto industrial standard for multichannel bioelectric data recordings. An interesting overview is given in: http://pub.ist.ac.at/~schloegl/publications/schloegl2009.pdf

However, the European Data Format (EDF and its extension EDF+, http://www.edfplus.info/) as well as the General Data Format (GDF, http://arxiv.org/abs/cs/0608052) are probably the most common data formats for the exchange of bioelectric data recordings today. The BIOSIG project offers an open source library for reading / writing those formats, see: http://biosig.sourceforge.net/documentation.html

6.2.5 Physical Input and Sensor Abstraction Layer

Integrating physical sensors and input is essential for providing many specialized assistive technology. Especially the USB HID and Bluetooth HID (Human Input Device) profiles provide an easy way to integrate multimodal input in a convenient fashion. The Developer Space provides multiple components for interfacing components via these minimalistic widely adopted standards. The other widely adopted de facto standard for communication is serial line communication that today is mostly used as Virtual COM ports via e.g. USB CDC (Communications Device Class).

The Physical Web is a recently coined term by Google that shows, that web technology is moving further towards physical functionality. Another trend is the increased use of wearable devices for activity tracking that both expose big potential for accessibility improvements. One essential requirement for many applications is long battery life and small form factor. While specialized low power wireless protocols like ANT or Zigbee have been around to support such applications support in end user devices particularly for connecting external sensors has not taken off (First support Zigbee in a mobile phone was announced 2004). This has radically changed since the arrival of the Bluetooth 4 specification, which was adopted first by the iPhone. Particularly BT4 Low Enegy Generic Attribute Profile (BLE GATT) is widely adopted by both consumer grade hardware as well as personal device vendors. Even browser APIs like Google Chrome have direct support for BLE devices. Furthermore Bluetooth 4.2 will define a standardized way to access devices via an HTTP Gateway providing internet connectivity. GATT allows vendors to specify discoverable services via XML Specification. Although many sensors are often interfaced in very heterogeneous message formats, however, particularly to health and fitness applications a wide range of GATT services exist.

6.2.6 Authentication

Making security (particularly) authentication usable is an important part of practical accessibility. Many IT-products are made purposefully more complex and inaccessible in order provide a certain level of security. Particularly password based schemes cannot be considered

very usable or accessible. Single sign-on solutions based e.g. on the OAUTH Standard at least mitigate the need to remember multiple passwords, however, does not directly add to the usability.

Recently the FIDO Alliance has released specifications that define an open, scalable, interoperable set of mechanisms that reduce the reliance on passwords to authenticate users. The standards cover both password less authentication (UAF), that eliminate the need for passwords, e.g. by using biometrics, and second-factor (U2F), that add physical security to existing authentication mechanisms. The idea is that mechanisms can be used in an interchangeable fashion, so that user needs can be met without compromising security.

One particular area is WIFI access, which is often a prerequisite to access e.g. home automation or local service infrastructures. The Wi-Fi alliance standardized Wi-Fi Protected Setup to increase the ease of connection. It specifies PIN, push button, NFC and USB methods to connect to wireless networks. However, particularly the PIN method is subject to a specification flaw that leads to serious security issues.

6.2.7 Multimodal Interfaces

ETSI EG 202 048 presents guidelines for the design and use of multimodal symbols using a Design for All approach. It also provides a study of the needs and requirements for the use of multimodal symbols in user interfaces,with special emphasis on the requirements of people with disabilities and elderly people. ETSI EG 202 048 provides guidelines, good practice and case studies for the successful design and application ofmultimodal symbols using the Design for All approach. It will support the standardization process with respect to the use of multimodal symbols in modern user interfaces. Icons, symbols and pictograms are widely used components of user interfaces in ICT applications and services, e.g. for navigation, status indication and function invocation. Examples of such applications and services include information retrieval (e.g. Web sites), messaging (e.g. email and SMS), public services (e.g. public telephones and ATMs) and real time communication services (e.g. fixed and mobile telephony). The use of visual-only symbols in such applications and services creates temporary or permanent problems for all users. User groups most affected are blind and partially sighted people and users of mobile devices with limited visual display capabilities. All users can potentially benefit from the current and future possibilities of multimodal user interfaces. These interfaces combine communication channels, for example sound, graphics, video, speech, force and vibration. ETSI EG 202 048 does not deal with unimodal symbols, i.e. only visual symbols or only auditory "earcons", but with symbols that use at least two communication channels.

6.2.8 Haptic and Touch Interfaces

One specific part of ISO 9241 "Ergonomics of human-system interaction" is the specification of haptic interfaces. ISO 9241-910 provides a framework for understanding and communicating various aspects of tactile/haptic interaction. It defines terms, describes structures and models, and gives explanations related to the other parts of this ISO 9241 "900" subseries. It also provides guidance on how various forms of interaction can be applied to a variety of user tasks. It is applicable to all types of interactive systems making use of tactile/haptic devices and interactions. It does not address purely kinaesthetic interactions, such as gestures, although it might be useful for understanding such interactions.

Part 920 of ISO 9241 gives recommendations for tactile and haptic hardware and software interactions. It provides guidance on the design and evaluation of hardware, software, and combinations of hardware and software interactions, including: the design/use of tactile/haptic inputs, outputs, and/or combinations of inputs and outputs, with general guidance on their design/use as well as on designing/using combinations of tactile and haptic interactions for use in combination with other modalities or as the exclusive mode of interaction; the tactile/haptic encoding of information, including textual data, graphical data and controls; the design of tactile/haptic objects, the layout of tactile/haptic space; interaction techniques.

It does not provide recommendations specific to Braille, but can apply to interactions that make use of Braille. The recommendations given in ISO 9241-920:2009 are applicable to at least the controls of a virtual workspace, but they can also be applied to an entire virtual environment — consistent, in as far as possible, with the simulation requirements. (NOTE: It is recognized that some interactive scenarios might be constrained by the limitation that a real workspace is to be modelled in a virtual environment. Objects can be in suboptimal positions or conditions for haptic interaction by virtue of the situation being modelled)

ISO 9241-940 covers the evaluation of tactile / haptic interactions

Few standardized APIs for haptic feedback exist. The W3C Vibration API is specifically designed to address use cases that require simple tactile feedback only. Use cases requiring more fine-grained control are out of scope for this specification. This API is not meant to be used as a generic notification mechanism. Such use cases may be handled using the Notifications API specification. In addition, determining whether vibration is enabled is out of scope for this specification.

CHAI3D is an open source set of C++ libraries for computer haptics, visualization and interactive real-time simulation. CHAI 3D supports several commercially-available three-, six- and seven-degree-of-freedom haptic devices, and makes it simple to support new custom force feedback devices. CHAI 3D is especially suitable for education and research purposes,

offering a light platform on which extensions can be developed. CHAI 3D's support for multiple haptic devices also makes it easy to send your applications to remote sites that may use different hardware.

In short, CHAI 3D takes an important step toward developer-friendly creation of multimodal virtual worlds, by tightly integrating the haptic and visual representations of objects and by abstracting away the complexities of individual haptic devices.

7 Future Work

As stated in the beginning of this document, it represents work in progress that has been authored and will be authored in a collaborative way within the Developer Space. Standards, guidelines and technical specifications, as well as common interfaces are one essential component of the Prosperity4All ecosystem. While concrete actions will be taken to promote and advance standards (within the cross-cutting activities), the Developer Space will continue to link out to other activities and will be a starting point for Prosperity4All stakeholders when looking for standardized or efficient ways to connect to existing systems and to ensure wide accessibility to IT-Technology. One challenging part, that we have not yet completetely addressed in this regard, is to establish quality standards for the components, that will allow adopters to quickly assess existing solutions. At the same time the Developer Space needs to be inclusive as possible to address very special needs and also promote the pickup of not yet production level components. This maturing of technologies needs to be guided and APIs for standardized testing procedures need to be developed as core part of the architectural efforts within the Prosperity4All project. Particularly for actively curated content quality standards need to be developed as part of these efforts (similar to efforts such as http://openpreservation.org/technology/principles/software-maturity/) .

During the coming month we expect a high degree of convergence between different components of the Developer Space simply due to the fact that they will be used together in different implementations and that developers are committed to improving the usefulness and usability of their components that have been developed so far in very closed project contexts for a wide range of adopters. Standardization and especially standard adoption as a strategy will be an important aspect that we continue to consider on our way to a prosperous ecosystem and a lively Developer Space. As argued in large parts of this document, we particularly see Web standards as a priority in order to establish accessibility within a wide range of areas of daily life as Internet technology is continuing to prosper. In comparison to many other industry driven standards, Web standards are well received also by a wider range of enthusiastic programmers and prosumers that we like to bring to the accessibility domain within the Developer Space. Please continue to visit http://wiki.gpii.net/w/Developer_Space/Standards to follow and join our efforts and to contribute further to this document.