SmartGo – Secure Travel Management System

Design Document

Version 1.2 | October 2013

Revision History

Ver	Change Description	Changed in Sections	Date	Author	Reviewer

1.0	Initial Draft		17/9/2013	Ashalatha.C	Rama Rao Mundru, Atul Kumar
1.1	Changed	All	4/10/2013	Rama Rao M	Ashalatha C
1.2	Changed	All	10/10/2013	Rama Rao M,	Ashalatha C

References

Reader should read this document in conjunction with the following documents

No.	Document Name	Ver.	Location
1.			
2.			

Definitions, Abbreviation and Acronyms

The terms in use in the document are explained below

Acronym	Description
API	Application Programming Interface
JS	Java Script
PZH	Personal Zone Hub
PZP	Personal Zone Proxy
TLS	Transport Layer Security
UI	User Interface

Table of Contents

Problem Statement

Design and Implement a Secure Travel Management System using Webinos.

Employee Safety and Security is Paramount in a Corporate sector. Keeping employees secure on travel, can be a challenge for companies, as there is lack of proper Travel management system to track and check whether the employee has reached the destination safely or not.

The Biggest challenge for transport department of company is to control and avoid the risk of Crime or violence on employees during his or her travel in company's transport. In case of travel agency, it is a challenge to monitor the driver or to track whether the goods are delivered properly.

SmartGo is a Secure Travel Management System application based on Webinos framework which is a cross platform, multidevice and multiuser compatible. It primarily aims for providing security and convenience for a traveler.

SmartGo provides the solution to the above mentioned problem statement:

SmartGo uses Geolocation, chat, file transfer features of Webinos frame work to meet the Secure Travel Management System. This application can be installed on the host pc of the transport department of a company and the end user application installed on employees/drivers mobile or in car devices.

- **Geolocation feature:**

  Before an employee starts his journey, the transport department allocates a cab driver and sets the fixed route where the employee needs to go. Also the end user is shared with the details of driver (e.g. Webinos id etc.), so that the employee can track where the cab is currently located. This helps in planning his boarding time properly.

  Once an employee boards the cab, the transport department can trace whether the cab is going in fixed route or not which is already set. A continuous monitoring of the location can be done. If there is any change in direction of the path, application sends a wrong route alert to the transport department. This alert helps the department to act in case of emergency situation.

- **Chat feature**:

  SmartGo also provides convenience to the traveler with the facility of chat, where a person using this application can chat with any other person who is connected to his personal zone or any other personal zone. The Application uses the Messaging API to achieve this functionality.

  Chat functionality can help to update the status to the driver to traveler vise versa, suppose if a cab driver stops his journey, he can update the reason to the traveler through this SmartGo app itself, instead of opening new application for sending text message or calling from mobile.

- **File transfer feature:**

SmartGo provides the capability of sharing files when needed e.g. the transport department can share the details of employees and route list to the cab driver, it avoids taking print outs of the route list. So that it easy to pick up the employees at the specified locations while ongoing.

Security can be provided by using Geolocation feature and convenience can be provided by using chat and file transfer feature.

SmartGo use case diagram

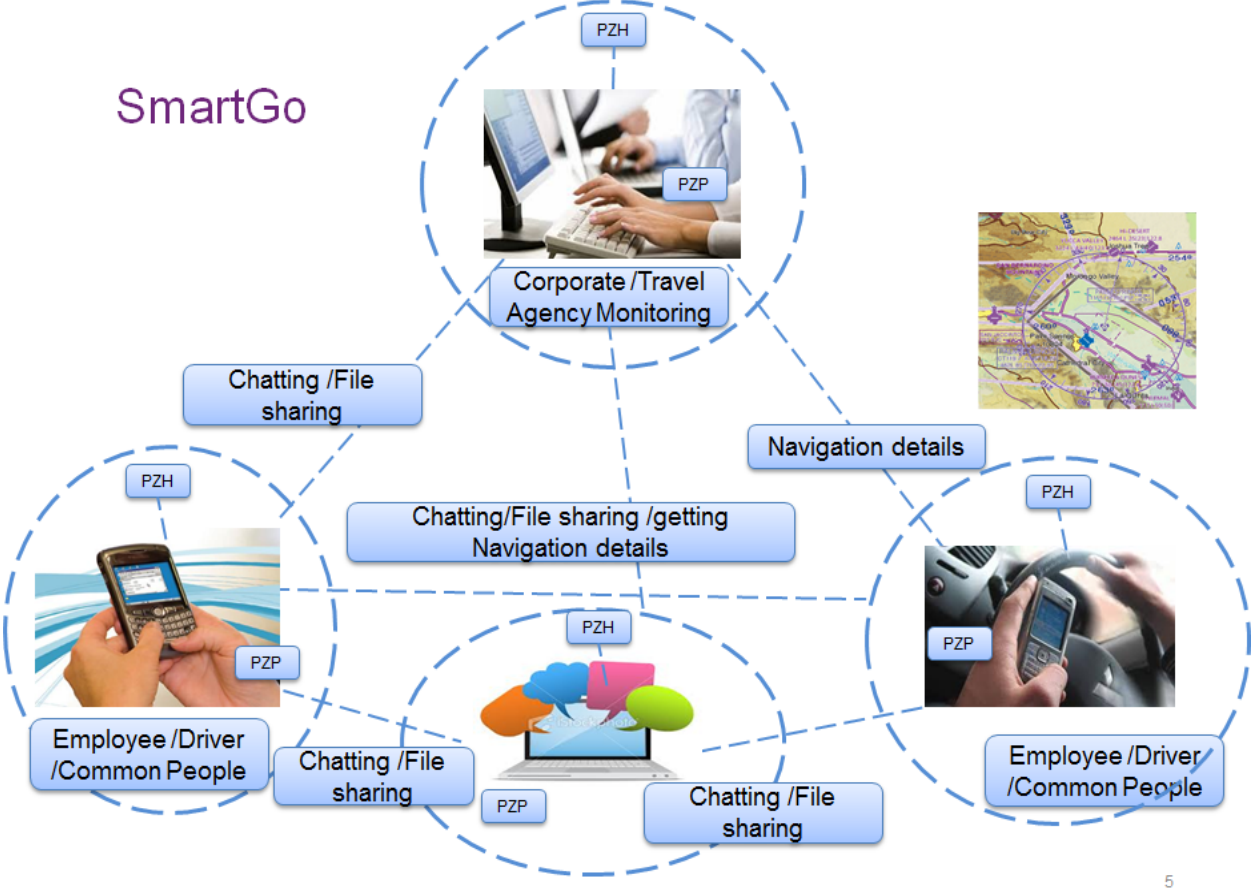

Figure 1: SmartGo use case diagram

Note: Please refer the appendix for understanding the Webinos concepts.

SmartGo use cases

3.1 Corporate offices

- **Use case:**

Transport department of company can use SmartGo application to provide security and safety to employees in travel.

- **Pre requirements:**

1. SmartGo user must need to connect to PZH.
2. SmartGo Host application must be installed on transport department PC.

3. SmartGo End user application must be installed on the cab drivers and travelers (employee) device.
4. Internet connection is required.

- **Use case scenario**

1. Once the employee raises the request for cab, the transport department shares the Webinos id's of the employee to the cab driver and vice versa by using **file sharing feature** of the SmartGo.
2. Employee and cab driver (PZP) has to connect each other through their Webinos ids.
3. Transport department sets the destination (fixed route) of the employee by using **Geolocation feature** of the SmartGo.
4. Once route of an employee is set, this route is called fixed route.
5. After the journey starts, if there is any deviation from the fixed route the wrong route alert will be sent to the transport department.
6. The transport department can take emergency actions to prevent the situation.
7. Once the destination is arrived, SmartGo pops up the destination reached safely with YES OR NO options.
8. If the employee reaches his/her destination safely he clicks the YES.
9. If he clicks YES it pops up DISCONNECT CONNECTION and sends info to the transport department that destination is reached safely and connection will be disconnected.
10. If he clicks NO it immediately sends an emergency situation to the transport department.

- **Post requirements:**

1. User must disconnect his/her connection with the cab driver and the transport department.

Note: For better safety and security purpose, user needs to click YES (destination reached safely) DISCONNECT only once he reached his/her home or where he wants to go.

3.2 Travel Agencies

- **Use case**:

Travel agencies can use SmartGo application for travelers and delivery of goods safely and securely.

- **Pre requirements:**
1. SmartGo user must need to connect to PZH.
2. SmartGo End user application must be installed on the vehicle driver's device or in vehicles.
3. SmartGo Host application must be installed on travel agency host PC.
4. Internet connection is required.
5. All the vehicles of a travel agency must be synchronized with the SmartGo host.

- **Use case scenario:**
1. Before the Passengers or Goods vehicle starts journey the agency's SmartGo application sets the vehicle's destination.
2. While in journey if the vehicle deviates from the actual path wrong route alert is sent to the travel agency (emergency actions can be taken).
3. If the vehicle reaches the destination safely the SmartGo app pops up the YES or NO option to on the receivers SmartGo APP (means the courier office or travel agency office).
4. If YES, the destination is reached safely and asks user to disconnect the connection and sends acknowledgement to the travel agency host application.

5. If NO, sends emergency action.

3.3 Schools and colleges

- **Use case**:

Schools and colleges can use the SmartGo application for student's safety and security in travel.

- **Pre requirements:**
  1. SmartGo user must need to connect to PZH.
  2. SmartGo End user application must be installed on the driver's device or in vehicle.
  3. SmartGo Host application must be installed on the transport department host PC.
4. Internet connection is required.
5. All the vehicles of a school or college must be synchronized with the SmartGo host.

- **Use case scenario:**
  1. Before the school or college vehicle starts journey the transport department SmartGo application sets the vehicle's destination.
  2. While in journey if the vehicle deviates from the actual path wrong route alert is sent to the transport department (emergency actions can be taken).
  3. If the vehicle reaches the destination safely the SmartGo app pops up the YES or NO option on the drivers SmartGo.
  4. If YES, the destination is reached safely and asks the user to disconnect the connection and sends an acknowledgement to the transport department's host application.
  5. If NO, sends emergency action.

3.4 Individual safety

- **Use case**:

SmartGo application can be used for an individual safety.

- **Pre requirements:**
  1. SmartGo user must need to connect to PZH.
  2. SmartGo End user application must be installed on his/her own personal devices.
  3. Internet connection is required.

- **Use case scenario 1:**
  1. An individual before starting his journey, has to set his/her destination on SmartGo application.
  2. While in journey if he/she deviates from the actual path wrong route alert is sent to the connected SmartGo app user (means he/hers care taker, so the emergency actions can be taken).
  3. If the user reaches the destination safely the SmartGo app pops up the YES or NO option on the user's SmartGo APP.

4. If YES, the destination is reached safely and asks the user to disconnect the connection, and sends acknowledgement to the connected user application.
5. If NO, sends emergency action.

- **Use case scenario 2:**
  1. SmartGo users can always be connected with their care takers, so that they are always monitored.

Note: Different SmartGo application is required for the different use case scenarios as mentioned above e.g. some applications requires host and end user application and some doesn't.

Requirements

- **Authentication and Connection:**

Requirement ID	Purpose	Dependencies
SGO_AC_Login_001	User login	
SGO_AC_Register_002	Register new user	
SGO_AC_ConnectReq_003	Sending connection request	User must be logged in to PZH and PZP
SGO_AC_AcceptReq_004	Accept the request	

SGO_AC_RejectReq_005	Reject the request	
SGO_AC_DeviceSyc_006	Getting sync with all the connected devices	
SGO_AC_Disconnect_007	Disconnecting from the device	

- **Geolocation:**

Requirement ID	Purpose	Dependencies
SGO_GE_GetLocation_008	Getting the current location (source)	
SGO_GE_SetLocation_009	Setting the destination	
SGO_SGO_GE_CheckLocation_010	Checking the location	Destination must be set
SGO_GE_WrongRouteAlert_011	To indicate the location mismatched	Location must be checked
SGO_GE_WatchLocation_012	Watching the location	
SGO_GE_StopWatchingTheLocation_013	Stop watching the location	

- **Chat:**

Requirement ID	Purpose	Dependencies
SGO_CH_OpenTextBox_014	Open textbox for writing message	
SGO_CH_TypeText_015	Writing text message	
SGO_CH_SendText_016	Sending message	
SGO_CH_ReceiveText_017	Receiving message	

- **File transfer:**

Requirement ID	Purpose	Dependencies
SGO_FT_BrowseFile_018	Searching the file to attach	
SGO_FT_AttachFile_019	Attaching file	
SGO_FT_SendFile_20	Sending file	File must be attached
SGO_FT_ReceiveFile_21	Receive file	
SGO_FT_OpenFile_22	Reading file	
SGO_FT_WriteFile_23	Writing into file	File must be opened
SGO_FT_CloseFile_24	Closing file	File must be opened

Webinos API’s

The following are the Webinos API’s used in application.

5.1 SmartGo Geolocation:

5.1.1 Discovery API

The Webinos Discovery API is used to discover the services.

- findservice(): This function finds the service.

5.1.2 W3C Geolocation API

The API exposes the functionality to interact with a satellite navigation service. It provides a method for handing over a point-of-interest to the service and monitoring if the navigation service is active.

The following are the functions used for Geolocation

- bindservice(): This function binds the service to the particular PZP.
- handle_geolocation(): This function is used to display a map with current position of the PZP with latitude, longitude.
- getCurrentPosition(); This function is used to get the current position of the PZP.
- showMap(): This function shows a map centered at particular latitude and longitude.
- watchPosition():This function is used to request repeated updates of location.
- clearWatch(): This function is used to clear the location updates.

5.2 SmartGo Chat:

Uses App2App Messaging API

The following are the functions used:

- findservice(): This function finds the Messaging service.
- bindservice(): This function binds the service to the particular PZP.
- createChannel(): This function creates a channel for communication.
- searchForChannels(): This function is used for searching the existing channels.
- connect(): This function is used to connect to the channel.
- send(): This function is used to send the chat message.

5.3 SmartGo File Sharing:

- W3C File API
  This API is used for representing file objects in web applications. As well as programmatically selecting them and accessing their data.

SmartGo High Level Architecture

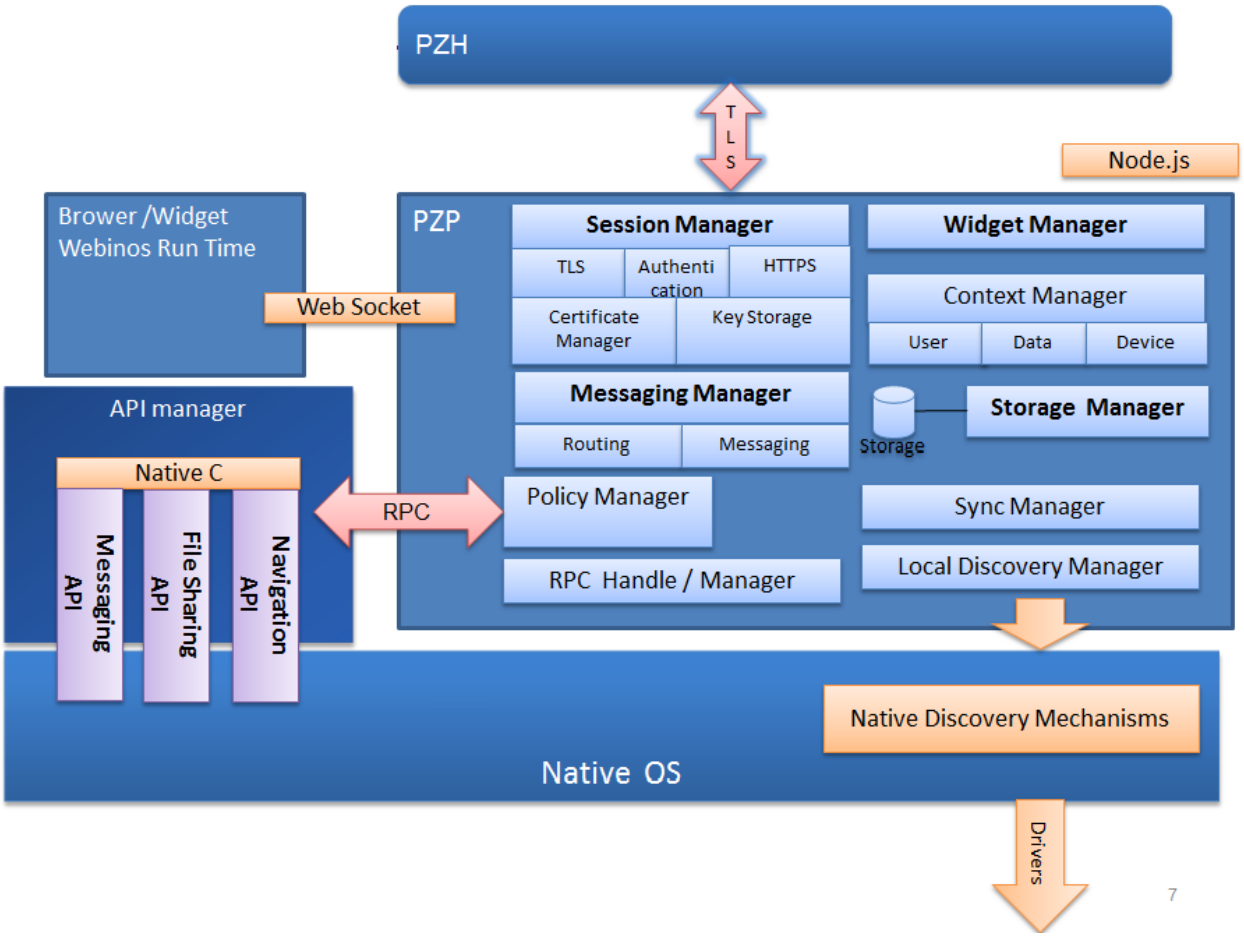

Figure 2 : High level architecture of SmartGo

The above diagram shows the Architecture of SmartGo. The PZP has the Session Manager which is used for Authentication. The Messaging Manager helps in routing the information to the PZH and the peers. The Context Manager hold the data related to user and the devices.

The API Manager manages the API's. SmartGo application uses the Navigation API, Messaging API and File Sharing API for its functionality.

The SmartGo application can be run on a browser or as a widget. The Webinos run time communicates with PZP through the web socket. The interaction between the API manager and the PZP is through Remote Procedure Calls. The Sync manager in the PZP is used for synchronization of all the devices.

Relation between various entities of SmartGo

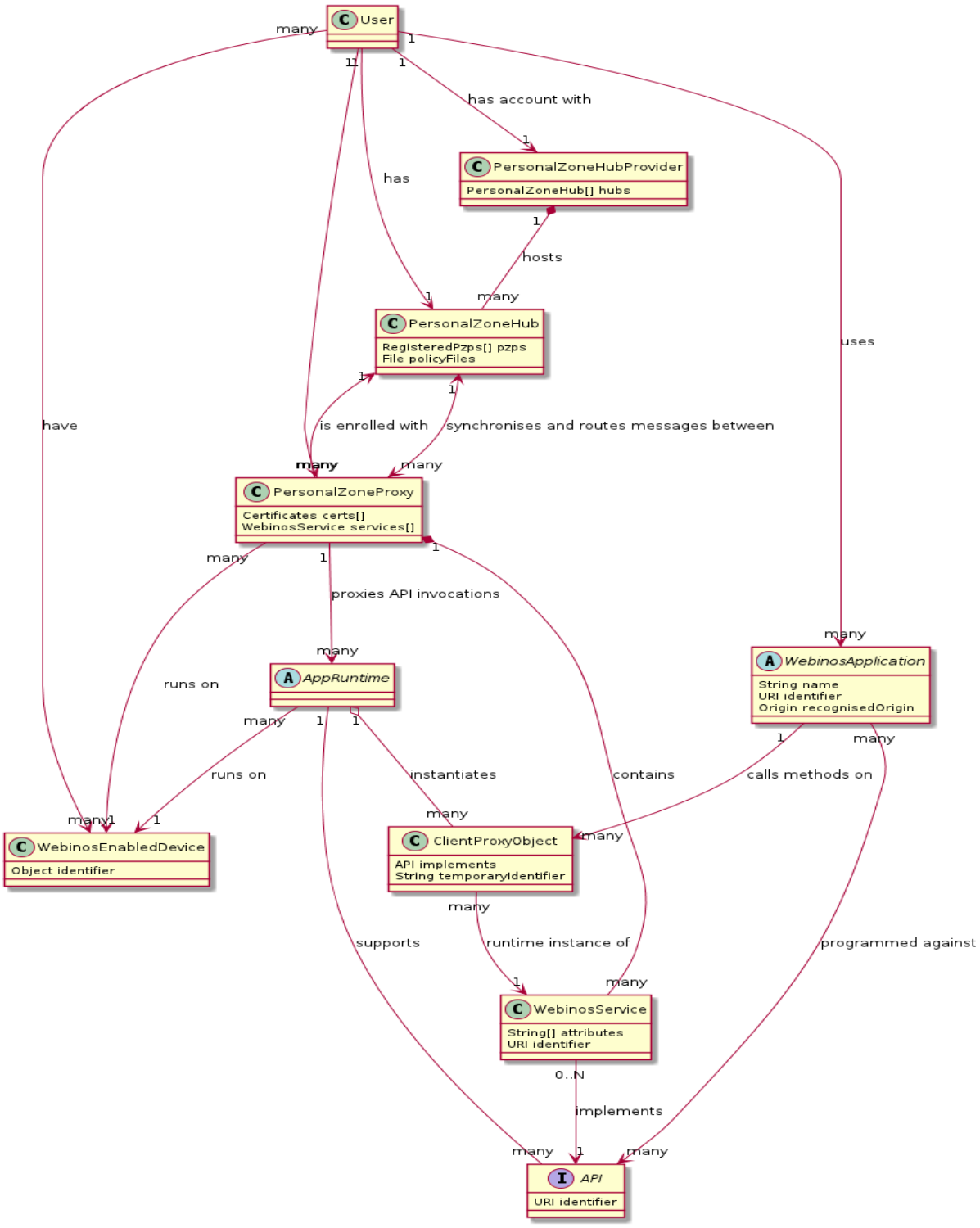

Figure 3: Relation between various entities of SmartGo

Key Considerations

The following are the Software dependencies for Webinos:

1. Visual Studio 2010 C++. Express Edition.
2. git for Windows
3. Python 2.7.3 - Install version 2.7.x

4. Node.js
5. OpenSSL: Need to install the VS2008 re-distributables.
6. Bonjour SDK for windows
7. GTK (that contains cairo library)
8. grunt 0.4 build system

Assumptions and dependencies

- **Dependency tools :**
  - Webinos frame work
  - OS : Windows, Linux
  - Extra tools : Visual C++ 2010, Python, GTK, OpenSSL, Node.js, Bonjour SDK for windows
  - Smart Phone
- **Assumptions:**
  - Users must have smart phones or Webinos enabled car
  - Host must have a PC

Appendix

10.1 Webinos

Today, the personal devices like Mobile, PC, and Tablet etc. are getting smarter and opportunities arise for sharing services, applications and data between them. While web technologies hold the promise of being a unifying layer, browsers lack functionality for supporting inter-device communication, synchronization and security. Webinos was designed to address this issue.

The main purpose of Webinos is to define and deliver an open source platform, which will enable web applications and services to be used and shared consistently and securely over a broad spectrum of connected devices. To achieve this, it defines and provides architecture and infrastructure to allow applications to run not only on a single device, but also across devices and domains.it is possible because Webinos is a web operating system, we can run Webinos platform independently.

Webinos is a cross-domain platform for secure web application delivery. These domains include mobile, PCs, home media (TVs), and in-car devices. It is specified as middleware installed on a selection of operating systems (OSs) on current devices to enable the consistent and secure web application user experience.

Webinos supports the following OSs and device platforms:

- Android 2.3.x tested with devices: Nexus S, Asus Transformer Prime, Samsung Galaxy S2, Sonyericsson Xperia Arc, Galaxy Note
- Microsoft Windows 7 SP, Windows 7, Windows XP tested with laptops: Vaio z11, Dell, Asus EeePC (1215N)
- Linux: Ubuntu 10.04 LTS, Slackware 13.1, 13.37, Mint, Fedora tested with devices: VMWare Player, Samsung, Asus EeePC (1215N) TV variants of Cocom Churchill 177, Acer Revo etc. vehicle variant of Pandaboard Rev. A3 with Ubuntu 11.10
- Mac OS X

The webinos platform includes not only a set of newly defined APIs to enhance current web application runtime environments, but also overlay network architecture to enable the Webinos specific features.

10.2 Basic Components of Webinos

Webinos is centered around the concepts on Personal Zones as a mean to organize personal devices (i.e. Mobile, tablet, desktop, smart TV, car unit device) and services running on them. This also includes personal services the user uses in the Cloud.

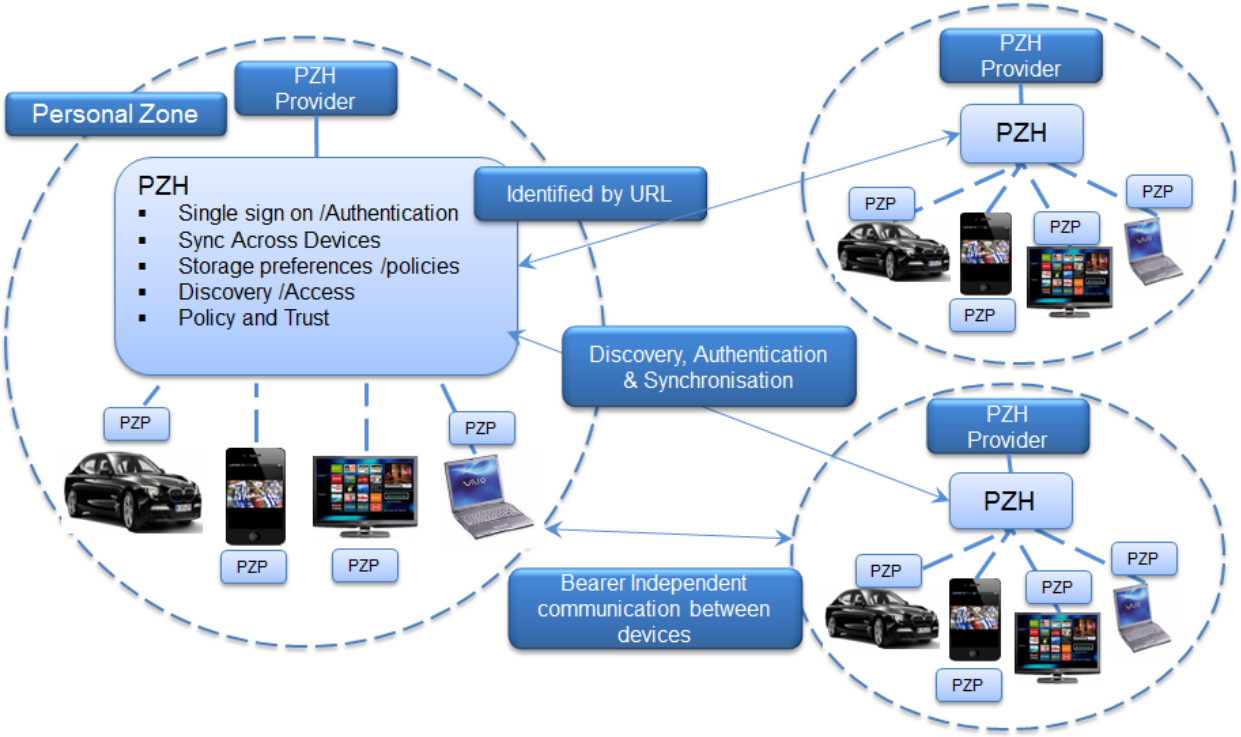

Figure 4: Elements of Webinos

The Personal Zone supports:

- Single sign-on, where the user is authenticated to a device and applications, and the device is authenticated to the Personal Zone. The architecture also allows for situations where the user is offline, e.g. when the user is away from home and currently unable to access the Internet.
- Shared model of the context: This covers users, device capabilities and properties, and the environment. It enables applications to dynamically adapt to changes, and to increase usability by exploiting the context.
- Synchronization across the devices in the zone: This includes support for distributed authentication, as well as personal preferences, and replication of service-specific data, e.g. social contacts, and appointments. Synchronization is essential for supporting offline usage.

- Discovery and access to services: This includes local discovery, e.g. of services exposed by the user's devices, whether connected through Wi-Fi, Bluetooth, or USB, as well as remote discovery for services exposed in the Cloud. The high level discovery API allows Web developers to search for all local services, or to filter by service type and context, or even to locate a named service instance. Remote discovery is based upon existing user names and Email addresses, resolving to a URI for a Personal Zone.

The Personal Zone is implemented on a distributed basis, consisting of a single Personal Zone Hub (PZH) and multiple Personal Zone Proxies (PZPs).Inside the Personal Zone each device installed with a PZP is connected to the PZH. The PZH is identified by a URL supporting Restfully APIs through JSON-RPC.

Normally the PZH is hosted by a PZH provider, which provides Cloud services. A PZH provider is a Service Provider that provides all Webinos services as an

operator. It provides PZH services to the end users, though the PZH may physically be installed on the user's premise. Also, users may act as their own provider.

The PZP entity resides with a user's device. A TLS session is set up between PZH and PZP, controlled by security policies.

PZH is part of the Personal Zone and supports access by the Personal Zone owner from other devices. E.g., when the owner is using a public computer in an Internet Cafe, the PZH enables him or her to access his or her Personal Zone's devices and services for the duration of a browsing session. It also enables access by others, subject to the policies that the owner has defined.

The PZH further provides support for discovering other PZHs based upon someone's full name, pseudonym, or Email address.

The Webinos Personal Zone Proxy runs locally on personal device. As a satellite proxy it acts like both a server (when talking to the end user on behalf the PZH) and a client (when talking to the PZH on behalf of the user and local services and applications). The PZP hosts Webinos services and applications, which makes it a server. The PZP acts in place of the PZH, when there is no Internet access to the central PZH server.

Conclusion

This document is prepared in the initial stage of the project. The content mentioned in this document could be removed or added in implementation.