

BONDI

BONDI – INTERFACES REQUIREMENTS

VERSION: Version 1.0

STATUS: Approved Release

DATE OF LAST EDIT: 26th May 2009

OWNER: OMTP Limited

CONTENTS

1	INTRODUCTION	8
1.1	DOCUMENT PURPOSE	8
1.2	BUSINESS RATIONALE	8
1.3	INTENDED AUDIENCE	8
1.4	CONVENTIONS.....	9
2	DOCUMENT SCOPE	11
2.1	INTERFACES IN SCOPE.....	11
2.2	DOCUMENT STRUCTURE	11
3	USE CASES.....	13
3.1	USE CASE 1: APPLICATION FUN PACK WIZARD.....	13
3.2	USE CASE 2: RINGTONE WIDGET	13
3.3	USE CASE 3: LOG NOTIFICATION WIDGET	13
3.4	USE CASE 4: PHOTO MANAGEMENT	13
3.5	USE CASE 5: AUCTION WIDGET	13
3.6	USE CASE 6: DRIVING DIRECTIONS WIDGET	14
3.7	USE CASE 7: SOCIAL NETWORKING.....	14
3.8	USE CASE 8: ADVANCED THEMES	14
3.9	USE CASE 9: CONTACTS	15
	3.9.1 <i>Description</i>	15
	3.9.2 <i>Flow</i>	15
3.10	USE CASE 10: MESSAGING	16
	3.10.1 <i>Use Case 10.1 - Frequent Messaging List</i>	16
	3.10.2 <i>Use Case 10.2 - Automatic SMS, EMail Sending</i>	16
	3.10.3 <i>Use Case 10.3 - Sending Web Content to Contact</i>	16
	3.10.4 <i>Use Case 10.4 - Autoblog</i>	17
3.11	USE CASE 11: CAPABILITY DISCOVERY.....	17
	3.11.1 <i>Description</i>	17
	3.11.2 <i>Flow</i>	17
3.12	USE CASE 12: CALLS.....	18
	3.12.1 <i>Use Case 12.1 - Click to Call</i>	18
	3.12.2 <i>Use Case 12.2 - Frequent Calls List</i>	18

3.13	USE CASE 13: TASK LIST	18
3.13.1	Use Case 12.3 - Family Shopping List.....	18
3.14	USE CASE 14: AGENDA.....	19
3.14.1	Use Case 14.1 - Appointment Weather.....	19
3.14.2	Use Case 14.2 - Take Me to The Meeting	19
3.14.3	Use Case 14.3 - Family Calendar	20
3.15	USE CASE 15: CONNECTIVITY - MANAGE CONNECTIONS FOR SPECIAL SITUATIONS (ROAMING, LOW BATTERY ETC).....	20
3.15.1	Description	20
3.15.2	Flow	20
3.16	USE CASE 16: DEVICE GALLERY.....	20
3.16.1	Description	20
3.16.2	Flow	21
3.17	USE CASE 17: LOCATION	21
3.17.1	Use Case 17.1 - Contact Map.....	21
3.17.2	Use Case 17.2 - Real Weather	22
3.17.3	Use Case 17.3 - 'Photo Upload Site Geo Tagging.....	22
3.17.4	Use Case 17.4 - TimeZone	22
3.17.5	Use Case 17.5 - Location Aware Searches	23
3.17.6	Use Case 17.6 - Where are my Friends?.....	23
4	GENERAL REQUIREMENTS	24
5	INVOKING APPLICATIONS.....	26
5.1	GENERIC FRAMEWORK.....	26
5.2	SPECIFIC APPLICATION TYPES	27
5.2.1	Telephony	27
5.2.2	Messaging.....	28
5.2.3	Browser.....	28
5.2.4	Camera	28
5.2.5	Media Player	29
6	MESSAGING	30
6.1	SCOPE	30
6.2	KEY ISSUE 1	30
6.3	KEY ISSUE 2	30
6.4	KEY ISSUE 3.....	31
6.5	REQUIREMENTS.....	31

7	MEDIA GALLERY.....	34
8	PERSISTENT DATA STORAGE.....	35
8.1	SCOPE	35
8.2	KEY ISSUES	35
8.3	TOP-LEVEL ORGANISATION OF FILE SYSTEM.....	36
8.3.1	<i>Interactive File/Directory Picking.....</i>	<i>37</i>
8.4	BINARY DATA SUPPORT	38
8.5	SYNCHRONOUS VS. ASYNCHRONOUS OPERATION	39
8.6	REQUIREMENTS.....	39
9	DEVICE STATUS AND PROPERTIES	42
9.1	SCOPE	42
9.2	ISSUE 1	42
9.3	ISSUE 2	42
9.4	ISSUE 3	42
9.5	ISSUE 4	42
9.6	ISSUE 5	42
9.7	REQUIREMENTS.....	43
9.7.1	<i>Battery.....</i>	<i>43</i>
9.7.2	<i>Connectivity</i>	<i>43</i>
9.7.3	<i>User Interaction Settings.....</i>	<i>43</i>
9.7.4	<i>Memory.....</i>	<i>44</i>
9.7.5	<i>Device.....</i>	<i>44</i>
9.7.6	<i>Operating System</i>	<i>44</i>
9.7.7	<i>Web Runtime Environment</i>	<i>45</i>
9.7.8	<i>Web Browser</i>	<i>45</i>
9.7.9	<i>Java Runtime Environment.....</i>	<i>45</i>
9.7.10	<i>OMA Push Client.....</i>	<i>45</i>
9.7.11	<i>OMA Multimedia Messaging Client.....</i>	<i>45</i>
9.7.12	<i>Certificates</i>	<i>45</i>
9.7.13	<i>Camera</i>	<i>46</i>
9.7.14	<i>Input/Output</i>	<i>46</i>
10	COMMUNICATION LOG.....	47
10.1	SCOPE	47
10.2	KEY ISSUE 1 - EXTENSIBILITY.....	47

10.3	KEY ISSUE 2 - SECURITY AND PRIVACY CONSIDERATIONS.....	47
10.4	REQUIREMENTS.....	48
11	PERSONAL INFORMATION	49
11.1	SCOPE	49
11.2	KEY ISSUE 1: MANAGING DATA FROM DIFFERENT SOURCES...	49
11.3	KEY ISSUE 2: FIELD SUPPORT.....	49
11.4	EVENT MANAGEMENT	50
11.5	CONTACT MANAGEMENT	51
11.6	TASK MANAGEMENT	52
12	LOCATION.....	54
12.1	SCOPE	54
12.2	KEY ISSUES	54
12.3	REQUIREMENTS.....	54
13	USER INTERACTION	57
13.1	SCOPE	57
13.2	KEY ISSUE 1.....	57
13.3	KEY ISSUE 2.....	57
13.4	KEY ISSUE 3	58
13.5	KEY ISSUE 4	58
13.6	KEY ISSUE 5	58
13.7	REQUIREMENTS.....	59
14	CAMERA.....	62
14.1	SCOPE	62
14.2	ISSUE 1	62
14.3	ISSUE 2	62
14.4	REQUIREMENTS.....	63
15	SYSTEM EVENTS	65
15.1	REQUIREMENTS.....	65
16	APPLICATION SETTINGS	67

16.1	SCOPE	67
16.2	KEY ISSUE 1	67
16.3	KEY ISSUE 2	68
16.4	KEY ISSUE 3	69
16.5	KEY ISSUE 4	70
16.6	KEY ISSUE 5	71
16.7	REQUIREMENTS.....	72
17	DEFINITION OF TERMS.....	73
18	ABBREVIATIONS	75
19	REFERENCED DOCUMENTS.....	77

The information contained in this document represents the current view held by OMTP Ltd. on the issues discussed as of the date of publication.

This document is provided “as is” with no warranties whatsoever including any warranty of merchantability, non-infringement, or fitness for any particular purpose. All liability (including liability for infringement of any property rights) relating to the use of information in this document is disclaimed. No license, express or implied, to any intellectual property rights are granted herein.

This document is distributed for informational purposes only and is subject to change without notice. Readers should not design products based solely on this document.

© 2009 OMTP Ltd. All rights reserved. OMTP and OMTP BONDI are registered trademarks of OMTP Ltd.

1 INTRODUCTION

1.1 DOCUMENT PURPOSE

A Web Runtime Environment (WRE) is a development and delivery platform that allows the creation of applications based on standard Web technologies, such as; HTML, SVG, CSS, JavaScript and AJAX.

On desktop computers, WREs offer the possibility to create applications quickly and deploy them easily across any kind of desktop computer platform.

In Terminals, different WREs have started to emerge trying to address the fragmented mobile market by allowing allow the same application to run regardless of the underlying mobile platform. Key to the growth of mobile WRE is the availability of suitable Application Programmable Interfaces (APIs) that allow application developers to access key Terminal features and functions such as contacts, messaging and telephony.

This document describes the minimum set of APIs and related requirements for a BONDI compliant Web Runtime.

1.2 BUSINESS RATIONALE

Many Terminals being delivered to the market offer a huge range of features that are accessible to developers through a set of APIs (e.g. messaging, contact list, IP connectivity etc.). Manufacturers and Operating System (OS) providers implement these features in different manners and hence they are offered inconsistently through their APIs across different platforms.

Therefore, developers willing to offer their applications in different Terminals typically need to develop and deploy different versions of their applications for each targeted platform.

WRE offers a platform in which the same service can be accessed by different clients, with low deployment effort. But in order to deliver this, it is essential that the issue of API fragmentation is addressed. If different companies implement Web runtimes in isolation and create a different set of proprietary interfaces, the most important principle of WRE will have been diluted and it shall become another fragmented platform.

1.3 INTENDED AUDIENCE

- Contributors to reference implementation.
- SDOs (e.g. W3C, OAA etc.)
- Application developers.
- OS vendors.
- Browser suppliers

- Original Equipment Manufacturers (OEMs).
- Operators.

1.4 CONVENTIONS

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119 [1].

- **MUST:** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- **MUST NOT:** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- **SHOULD:** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT:** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.
- **MAY:** This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option **MUST** be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option **MUST** be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

The requirements within this document are uniquely identified using the following format:

IFC-####.#, where:

- IFC is the acronym used to identify the subject of this OMTP public working draft document (i.e. BONDI Interfaces)
- #### is a 4 digit number that identifies the requirement (e.g. 0020) and which is to be unique within the document.

- .# are numbers that indicate sub-requirements (e.g. 00020.1 & 00020.2 which would be sub-requirements of 00020 and 00020.1.1 & 00020.1.2 which would be sub-requirements of 00020.1).

2 DOCUMENT SCOPE

This document focuses on the definition of detailed requirements for the programming interfaces that should be offered in a mobile WRE allowing the use of Terminal resources. These programming interfaces are called "BONDI Interfaces" or "BONDI APIs". The document covers requirements that clearly define the expected functionality to be available to developers through these interfaces and makes recommendations on how the requirements should be implemented. The definition of an API for which all the methods and properties are specified is out of the scope of this document as this will be delivered as part of the BONDI Reference Implementation(s) to be developed after the completion of this document.

The requirements in this document shall apply to scenarios where the Web Application runs in the Terminal (i.e. an installed Widget) and where a Web Application is hosted on a web server (i.e. BONDI capable Web Page).

The term Web Application shall be used generically in this document to cover a wide range of applications such as installed Widgets, BONDI capable Web Pages and Hybrid Applications.

The definition of requirements relating to the application packaging model, security model and the policies that manage access to a programming interface is out of the scope of this document, they are described in the Architecture and Security Requirements Document [2].

2.1 INTERFACES IN SCOPE

This document focuses on the BONDI Interfaces that shall enable access to, and use of Terminal specific resources by WREs. These interfaces are currently not supported by the Web User Agents for desktop computers, set-top boxes and Mobile Terminals.

This document does not try to define the requirements for each and every interface available to application developers, as the number of such interfaces is huge.

However, this document does not preclude the implementation of additional interfaces in order to extend the identified minimum functionality. However, any additional interfaces shall fit into the Architecture and Security Model defined in [2].

2.2 DOCUMENT STRUCTURE

Chapter 3 describes the use cases that have been used as the main input for the identification of requirements.

Chapter 4-16 define the high and mid level requirements that describe the functionality the WRE shall expose through the BONDI Interface to Web Applications. The requirements included in these chapters are not intended to be linked on a one-to-one basis to specific methods or properties in BONDI implementations (e.g. a method in one API could fulfil two requirements simultaneously).

High Level requirements are intended to specify the use case or the broad functionality that BONDI is pursuing. On the other hand, mid-level requirements provide a deeper level of detail on how those use case should be implemented (e.g. API parameters, possible return values, method of operation). From a compliance point of view, BONDI is looking for compliance with the mid level requirements as they provide a more granular point of view on the functionalities required (i.e. an implementation fulfilling the High Level requirements but not the mid-level ones that would not be BONDI compliant).

3 USE CASES

The following use cases give examples of how Widgets and BONDI capable Web Pages use Terminal specific interfaces. The use cases do not necessarily reflect the final set of BONDI Interfaces to be agreed.

3.1 USE CASE 1: APPLICATION FUN PACK WIZARD

Matt buys a new Terminal from his operator. When he starts up the Terminal for the first time, the set-up wizard asks if he wants to install the operator's "fun pack" of applications. He clicks yes, and opens a catalogue of applications. He clicks to select the applications he wants from the fun pack, and they are downloaded, installed, and added to the Mobile Terminal's home screen automatically.

3.2 USE CASE 2: RINGTONE WIDGET

Kate is a teenager, and loves chart music. Most weeks, she wants a new ringtone. When she goes into the Terminal's ringtone menu to change it, she clicks on "top 10" and can immediately choose from the current Top 10 chart hits in her country, listening to a preview of each before choosing to buy one.

3.3 USE CASE 3: LOG NOTIFICATION WIDGET

Roman is a new father who is often away from home. He has subscribed to his wife's "baby blog". His Terminal's home screen tells him when his wife has uploaded a new comment, picture or video to her blog. When he opens his video album, the latest video clip is already downloaded for him to enjoy.

3.4 USE CASE 4: PHOTO MANAGEMENT

Morand is a keen photographer. He has several hundred photographs organised into albums on his Terminal. When he wants to share an album, he simply chooses "share this album" on his Terminal and the images are uploaded to his remote web album. He can also organise his photos into slideshows on his Terminal, which his friends can view on the web.

3.5 USE CASE 5: AUCTION WIDGET

Becky buys many of her clothes from an auction web site. She has several designers she likes, and has saved searches for new items in her sizes. When she has time, she can browse all the newly listed items on her Terminal. If she chooses to make a bid, her bidding and watching list is immediately accessible from her Terminal's home screen. If she is outbid, an alert appears on her home screen immediately, and she can bid back with one click based on the bidding parameters she has previously entered into the Terminal.

3.6 USE CASE 6: DRIVING DIRECTIONS WIDGET

Gary works for a home installations company. His job is to visit customers in their homes and fix their issues with cable TV or broadband. His appointments are scheduled by a dispatcher in a call centre. Whenever Gary looks at his calendar during the day, he can see all his upcoming appointments, with addresses. When he is ready to set off to the next appointment, he can click on the address to instantly get driving directions and a map that has been downloaded to his Terminal while he has been working.

3.7 USE CASE 7: SOCIAL NETWORKING

Jenna is browsing the 'Social Networking Site' page of her friend Alice. Jenna discovers there are a set of new functionalities that allow her to communicate with Alice:

- Input text in a combo box and send an SMS to Alice without knowing Alice's phone number.
- Start a voice call with Alice without knowing Alice's phone number (through the native dialler application).
- Download Alice's images on the 'Social Networking Site' page to her Terminal.
- Upload a picture from her Media Gallery to Alice's 'Social Networking Site' gallery.
- Add content to Alice's 'Social Networking Site' wall.
- Download the contacts of any of Alice's 'Social Networking Site' buddies to the Terminal contact list.

Jenna wishes to add content from her Terminal to her 'Social Networking Site' page. When she logs in to her 'Social Networking Site', her homepage allows her to perform the following actions:

- Upload images from her Media Gallery.
- Upload calendar entries to her 'Social Networking Site' calendar.
- Upload contacts to her 'Social Networking Site' buddy list.
- Upload her Terminal call log to the 'Social Networking Site' page.

3.8 USE CASE 8: ADVANCED THEMES

Joe is a teenager who loves comics. Whilst browsing a theme provider web page with his Terminal, he discovers he can download the new Arachnidman theme. He downloads the theme and it is installed seamlessly on his Terminal. This theme does not only consist of a colour scheme and wallpaper, it is a Widget that is displayed on the Terminal's idle screen and includes:

- Links to launch native applications such as:
-

- Messaging Application.
- Telephony Application.
- Multimedia Player.
- Media Gallery.
- Items that are displayed as part of the background like:
 - Media Gallery Images (static one or a slideshow).
 - Last items in the call and message logs (they should be clickable in order to provide links to the native "Call Log Application" and "Messaging Application").
 - Events in Joe's calendar and tasks list.
 - News ticker including, for instance, the latest news related to Arachnidman.

The Widget is also notified whenever an incoming message or call is received so it can display animations whenever one of these events is received. The calls and messages are still managed by the native applications.

3.9 USE CASE 9: CONTACTS

3.9.1 DESCRIPTION

This use case describes how a Widget can remind a user of upcoming birthdays or anniversaries and suggest a gift.

3.9.2 FLOW

- 1 The widget gets launched (or goes foreground) whenever an upcoming birthday is detected.
- 2 The widget retrieves the contact details (sex, age, home address etc.,) that are stored in the Terminal and (optionally) queries the gift recommendation service to get a suggestion.
- 3 The user gets a notification and a suggestion for a gift and related information.
- 4 The user can click to browse the associated gift shop web site, send an e-mail, text or call to retrieve more information or buy it.

3.10 USE CASE 10: MESSAGING

3.10.1 USE CASE 10.1 - FREQUENT MESSAGING LIST

3.10.1.1 DESCRIPTION

This use case describes how a Widget can create a frequent messaging list depending on the week day, the location, time of day etc.

3.10.1.2 FLOW

1. The Widget runs in the background (i.e. it is invisible) and waits to be triggered by an incoming call.
2. The Terminal receives a voice call.
3. The Widget is triggered.
4. The Widget retrieves and stores the data about this call and uses it to provide a frequent call list on demand depending on the data.
5. The user can use this list to start a new call or send a message.

3.10.2 USE CASE 10.2 - AUTOMATIC SMS, EMAIL SENDING

3.10.2.1 DESCRIPTION

This use case describes how a Widget can automatically know when the user is busy and reject any incoming call and send an apology message to the caller.

3.10.2.2 FLOW

1. The Widget runs in the background and detects any incoming call.
2. If a call is detected the Widget checks whether a meeting is taking place by recognising that the Terminal is running in meeting mode.
3. The Widget rejects the call if the above is true. It attempts to determine whether the user is already in the contacts list based on the calling ID. If that attempt succeeds the Widget will determine the name and assemble a personalised message which can be send to the caller via SMS and/or email
4. The user can create a VIP list of people whose calls are never rejected.

3.10.3 USE CASE 10.3 - SENDING WEB CONTENT TO CONTACT

3.10.3.1 DESCRIPTION

This use case describes how a BONDI Capable Web Page can provide a user with a way to send content (images, urls, text etc.,) to any other contact.

3.10.3.2 FLOW

1. The user browses to the BONDI Capable Web Page.
2. The user finds a picture inside a news article that he wants to forward to a friend.
3. The user selects the picture and activates the “Forward to Friend” messaging function.
4. The user chooses a locally stored contact’s phone number to send the picture by MMS to this phone number.
5. The user activates the content submission function.
6. The BONDI Capable Web Page sends an MMS containing the picture selected by the user to the phone number.

3.10.4 USE CASE 10.4 - AUTOBLOG**3.10.4.1 DESCRIPTION**

This use case describes how a Widget can be used to automatically create a blog containing all the messages and calls.

3.10.4.2 FLOW

- 1 The user clicks on the Widget and it runs in background detecting incoming and outgoing messages and calls.
- 2 Whenever an incoming or outgoing message or call is detected the Widget retrieves the data (date, time, contact etc.,) and sends it to the user’s personal blog.
- 3 The user can connect to their blog to check their communications history by means of a Widget or an Active Web Page.

3.11 USE CASE 11: CAPABILITY DISCOVERY**3.11.1 DESCRIPTION**

This use case describes how a Web Application (i.e. a Widget or BONDI capable Web Page) shall be able to check which scriptable APIs are available on a Terminal, to prevent malfunction.

3.11.2 FLOW

- 1 Upon Web Application installation the Widget Manager checks its configuration file to confirm which scriptable APIs need to be present on the Terminal.
- 2 It is checked if the Terminal WRE supports the set of required APIs. If a critical API is missing installation fails.
- 3 If there are no critical APIs missing the installation may proceed.

- 4 During run time the Web Application should be able to check if any optional API is present and run accordingly to that.

3.12 USE CASE 12: CALLS

3.12.1 USE CASE 12.1 - CLICK TO CALL

3.12.1.1 DESCRIPTION

This use case describes how a Web Application adds a "click to call feature" to any advert or banner.

3.12.1.2 FLOW

1. The user clicks on the Widget or browses to the BONDI Capable Web Page.
2. The Web Application shows a banner or an ad.
3. The user clicks on the banner because she is interested in the offer.
4. The Web Application sets up a call to the number which is associated with the banner.

3.12.2 USE CASE 12.2 - FREQUENT CALLS LIST

3.12.2.1 DESCRIPTION

This use case describes how a Widget creates a frequent calls log depending on the week day, the location, the time etc

3.12.2.2 FLOW

1. The Widget runs in the background and monitors any incoming call retrieving location.
2. The Widget stores details for the call, calling number, duration, outgoing called numbers and uses this information to provide a frequent call list depending on the data.
3. The user can use this list to start a new call.

3.13 USE CASE 13: TASK LIST

3.13.1 USE CASE 12.3 - FAMILY SHOPPING LIST

3.13.1.1 DESCRIPTION

This use case describes how a Widget provides a family with a common shopping list.

3.13.1.2 FLOW

1. The user launches the Widget on the Terminal.
2. The Widget connects to a Family shopping list back-end service to update the task list in the Terminal.
3. The user can use the Widget to locally manage items in her own shopping list that will be uploaded to the Family shopping list back-end service by the Widget afterwards.

3.14 USE CASE 14: AGENDA**3.14.1 USE CASE 14.1 - APPOINTMENT WEATHER****3.14.1.1 DESCRIPTION**

This use case describes how a Widget provides a user with weather forecast information for a given appointment (location, date and time).

3.14.1.2 FLOW

1. The user launches the Widget
2. The Widget retrieves the list of appointments from the local PIM database along with their associated information such as location, date and time etc., and displays the list of appointments.
3. The user selects an appointment from the list and clicks on "Get Forecast".
4. The Widget sends the location and date and time to the Weather web service, retrieves a forecast and shows it.

3.14.2 USE CASE 14.2 - TAKE ME TO THE MEETING**3.14.2.1 DESCRIPTION**

This use case describes how a Widget provides a user with maps and information to get to an appointment location.

3.14.2.2 FLOW

1. The Widget is automatically launched a pre-determined time before the start of a meeting.
2. The Widget retrieves the location information for the appointment and the Terminal's actual location.
3. The Widget sends location data to a Mapping and Routing web service that calculates the shortest route. The Mapping and Routing web service responds with map that highlights the route.
4. The Widget sends location updates to the Mapping and Routing web service to update the map whenever the current location changes a certain amount.

5. The user closes the Widget whenever she gets to the meeting location.

3.14.3 USE CASE 14.3 - FAMILY CALENDAR

3.14.3.1 DESCRIPTION

This use case describes how a Widget provides a family with a common calendar merging all the family members' appointments.

3.14.3.2 FLOW

1. The user launches the Widget
2. The Widget retrieves the family's appointments from a remote repository.
3. The Widget displays the user's locally stored personal appointments together with the rest of the family appointments in a calendar view.
4. The user can locally and remotely view, create, edit or delete any appointment.

3.15 USE CASE 15: CONNECTIVITY - MANAGE CONNECTIONS FOR SPECIAL SITUATIONS (ROAMING, LOW BATTERY ETC)

3.15.1 DESCRIPTION

This use case describes how a Widget uses the system status (e.g. active connection type, roaming, battery etc) to avoid unexpected costs.

3.15.2 FLOW

1. The Widget runs in the background and monitors the system status relating to the Terminal's wireless connection and the battery's remaining power.
2. As soon as a wireless connection is established, the Widget checks for audio track updates on a remote server.
3. If the Terminal has enough remaining power, the Widget asks the user to download the latest audio tracks. If the battery has insufficient power to complete the task, the Widget notifies the user about the new audio tracks but warns him/her not to download them due to the low battery level.

3.16 USE CASE 16: DEVICE GALLERY

3.16.1 DESCRIPTION

This use case describes how a Widget uploads picture content to a social network.

3.16.2 Flow

1. The user launches the Widget
2. The user selects the function to upload a new picture for her avatar to the remote social network server.
3. The Widget lists all the files within the Media Gallery or gives the user the option to take a new picture using the built-in camera.
4. The user selects a locally stored picture.
5. The Widget reads some file information from this picture which it uses to preset some input fields in the picture metadata input form.
6. The user edits the picture's metadata.
7. The user starts the upload and the Widget sends the selected file with the metadata to the remote server.

3.17 USE CASE 17: LOCATION**3.17.1 USE CASE 17.1 - CONTACT MAP****3.17.1.1 DESCRIPTION**

This use case describes how a Widget or a BONDI Capable Web Page can provide a user with maps and information on where each one of her contacts live and route information to get there.

3.17.1.2 FLOW

The user clicks on the Widget or browses the BONDI Capable Web Page.

The Web Application retrieves the contact list along with their home address information.

The user selects a contact and clicks on "Get Map".

The Web Application geo-locates the contact home address and retrieves and shows a map for it using a mapping web service.

The user clicks on "Directions" to get a route linking its real location and its contact home address.

The Widget retrieves the Terminal real location.

The Web Application sends the Terminal location along with the contact home address to the Routing web service to get a route and display it.

3.17.2 USE CASE 17.2 - REAL WEATHER

3.17.2.1 DESCRIPTION

This use case describes how a Widget provides a user with weather information for their current location.

3.17.2.2 FLOW

1. The user launches the Widget.
2. The Widget reads the current Terminal's location.
3. The Widget sends a request to a weather service on the web using location as one of the criteria; receives the forecast and displays it.

3.17.3 USE CASE 17.3 - 'PHOTO UPLOAD SITE GEO TAGGING

3.17.3.1 DESCRIPTION

This use case describes how a Widget or a BONDI Capable Web Page can be used to take pictures or browse the gallery and upload content to photo upload sites automatically adding location information.

3.17.3.2 FLOW

The user clicks on the Widget or browses the BONDI Capable Web Page.

The Web Application displays a list of pictures from the Terminal Gallery and provides the user with the option to launch the camera and take a picture.

The user selects one of the gallery pictures or takes a new picture and clicks on "Upload to 'Photo Upload Site'".

The Web Application uploads the picture along with location information relating to where it was taken or where the terminal was when it was uploaded.

3.17.4 USE CASE 17.4 - TIMEZONE

3.17.4.1 DESCRIPTION

This use case describes how a Widget or a BONDI Capable Web Page can provide a user with information on local and home time zones whilst she is on the road.

3.17.4.2 FLOW

The user clicks on the Widget or browses to the BONDI Capable Web Page

The Web Application gets home location date, time and time zone from the terminal; it also gets current actual location information.

The Web Application uses location information to work out the local time zone by means of a geo location web service.

The Web Application works out the time shift between local and home and periodically updates both clocks.

3.17.5 USE CASE 17.5 - LOCATION AWARE SEARCHES

3.17.5.1 DESCRIPTION

This use case describes how a Web Application can provide a user with search results dependant on their real location which may be local hotels, pictures of local "points of interest" or possibly local services.

3.17.5.2 FLOW

The user clicks on the Widget or browses a BONDI Capable Web Page.

The user enters a criteria for a query (if needed, results could be delivered without user interaction) and clicks on "Search").

The Web Application adds location information to the query and provides with tailored results.

3.17.6 USE CASE 17.6 - WHERE ARE MY FRIENDS?

3.17.6.1 DESCRIPTION

This use case describes how a Widget or a BONDI Capable Web Page can provide a user with location information for a group of people.

3.17.6.2 FLOW

The user clicks on the Widget or browses to the BONDI Capable Web Page.

The Web Application periodically sends the user location information to the Friends location back-end service.

The user can then get a map with its own location and their friends' locations.

4 GENERAL REQUIREMENTS

The requirements defined in this section are applicable to all BONDI Interfaces. It is expected that any BONDI compliant implementation SHALL comply with these requirements.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0010	There MUST be a consistent error handling mechanism.	
IFC-0010.1		There MUST be a common base class for exceptions.
IFC-0010.2		Exceptions MUST be grouped by functionality (e.g. Security, Network, Terminal, BONDI etc.)
IFC-0010.3		There MUST be uniformly structured error codes within a single numeric space, but separating generic system-wide codes from ones specific to a particular API.
IFC-0010.4		Each exception MUST be linked to a unique name for referencing by the applications.
IFC-0020	APIs MUST be uniquely identified	
IFC-0020.1		Each API MUST be identified globally and uniquely by a namespace, so that meaningful actions can be taken to determine its availability, as well as to allow for runtime activation.
IFC-0030	Slow APIs MUST be implemented asynchronously	
IFC-0030.1		If an API (e.g. GPS receiver) is considered to be slow (in terms of response availability), it MUST provide an asynchronous way of operation.
IFC-0030.2		In case an asynchronous call fails because of system constraints (e.g. maximum number of simultaneous calls reached), a pre-defined error code MUST be returned.
IFC-0040	Functions MUST follow a consistent naming convention	
IFC-0040.1		camelCase naming convention MUST be used. I.e. the first letter of each word constituting the function name MUST be capitalized except for the first word
IFC-0040.2		The existing conventions used throughout various APIs MUST be reused.
IFC-0050	The BONDI coding style guide MUST be followed	
IFC-0050.1		All the APIs MUST follow a coding style guide defined by OMTP BONDI or other bodies.
IFC-0060	APIs MUST be discoverable	

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0060.1		It MUST be possible to discover whether objects and methods are supported.
IFC-0060.2		If an API is not supported, the corresponding objects and methods SHALL have an undefined value.
IFC-0060.3		In particular, if an application tries to use an unsupported object or method, an exception MUST be thrown.
IFC-0070	APIs MUST be functionally grouped	
IFC-0070.1		The APIs MUST be divided into sub-groups according to their relevance and commonality in desktop and mobile environments (these are subject to interpretation).
IFC-0070.3		Objects and functions defined SHALL be under a global object (e.g. omtp).
IFC-0070.4		Objects and functions SHALL be defined in such a manner that migration to extend the "Widget" object is possible (e.g. omtp.location to widget.location).

5 INVOKING APPLICATIONS

For many use cases it is very important to permit Web Applications to start another application. It is assumed the underlying platform allows launching multiple different applications at the same time. This section defines the requirements on a generic interface to invoke another application. Additionally, for those applications deemed as the most important ones, a set of specific requirements are defined.

Invoking applications is potentially a very sensitive resource for an end-user and necessary precautions need to be taken to manage access to this interface. The potential for abuse is very high as any application, regardless of its trust level, can be invoked through this mechanism.

5.1 GENERIC FRAMEWORK

This framework provides an interface that allows Web Applications to invoke any other application on a Terminal. It is necessary to uniquely identify applications across all Application Execution Environments. Additionally, an application could perform different actions when it is started depending on the context (e.g. parameters, selected options etc).

The URI schemes and MIME types that can be associated with applications should not be limited by BONDI, i.e. if the underlying platform supports the registration of applications as the default handler for arbitrary URI schemes and MIME types, the BONDI WRE should be capable of disclosing the association of the application to the registered URI scheme.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0080	A Web Application MUST be able to launch an application.	
IFC-0080.1		It MUST be possible to identify a specific application through an Application ID. NOTE: Application IDs uniquely identify an application after it has been installed in the terminal. Application IDs are platform specific and vary across different platforms and terminals.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0080.1A		<p>It MUST be possible to invoke an application through URI schemes (e.g. as the default handler for a URI scheme [3]), e.g.:</p> <ul style="list-style-type: none"> • http/https • sip/sips • tel • mailto • sms • mmsto • wtai: <p>NOTE: Different applications may be able to handle the same content. Resolving the target application linked to the URI is the responsibility of the underlying platform.</p>
IFC-0080.2		It MUST be possible to specify a set of parameters that are forwarded to the application to be launched.
IFC-0080.3		It MUST be possible to launch any type of application in whichever Application Execution Environment (AEE) it is executed in (e.g. Java, Native, Widgets).
IFC-0090	A Web Application MUST be able to discover the installed (i.e. registered) applications and their Application IDs available in the Terminal.	
IFC-0090.1		The full list of applications and Application IDs and other application related metadata installed on the Terminal MUST be returned.
IFC-0090.1A		It MUST be possible to identify the application registered on the Terminal as the default handler for a URI scheme.
IFC-0090.1B		It MUST be possible to identify the application registered in the Terminal as the default handler for a MIME type.

5.2 SPECIFIC APPLICATION TYPES

As a minimum, all the default handlers related to URI schemes for the application types listed in this section must be able to be launched through the generic framework described in Section 5.1.

5.2.1 TELEPHONY

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0100	A Web Application MUST be able to launch the Telephony Application.	
IFC-0100.1		It MUST be possible to specify the Call Type (e.g. Voice, Video etc.).

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0100.2		It MUST be possible to specify the destination URI (e.g. tel, sip etc.).

5.2.2 MESSAGING

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0110	A Web Application MUST be able to launch the messaging application to create a message.	
IFC-0110.1		Where supported by a related URI scheme, it MUST be possible to specify various message components (such as type, destination, subject etc..) to be reflected in the messaging application UI..

5.2.3 BROWSER

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0120	A Web Application MUST be able to launch the browser application.	
IFC-0120.1		It MUST be possible to specify the destination URL.
IFC-0120.2		It SHOULD be possible to specify the Connection Profile to be used (e.g. APN, credentials and proxy in the case of 2G/3G connections).

5.2.4 CAMERA

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0130	A Web Application MUST be able to launch the native camera application.	
IFC-0130.1		It SHOULD be possible to specify the camera to be used (e.g. rear or front).
IFC-0130.2		It SHOULD be possible to specify the media type (e.g. still image, video etc.).
IFC-0130.3		It SHOULD be possible to specify the resolution of the media.

5.2.5 MEDIA PLAYER

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0140	A Web Application MUST be able to launch the native media player.	
IFC-0140.1		It MUST be possible to specify the content to be rendered or played.

6 MESSAGING

6.1 SCOPE

The possibility to launch the native messaging application is part of the requirements defined in Section 5 (Invoking Applications), these requirements state that a Web Application must be enabled to launch the messaging application and populate it with data e.g. message type, body and recipients, leaving the final decision to send or store the message to the user by means of the native application.

There may be use cases for which it would be more efficient to enable Web Applications to programmatically deal with messaging capabilities independently of the native messaging application. For example, Web Applications, using messaging technologies (SMS, MMS or e-mail) as a means of sending application data to the network.

Enabling Web Applications to manage these capabilities by directly interfacing with the native application, removing user controls raises some threats that must be addressed by the terminals security framework. This is one of the reasons to provide this functionality in a dedicated interface, isolated from other related interfaces such as Communication Log which raises different risks and therefore needs different security policies.

Messaging requirements have been kept intentionally simple since the final goal of the BONDI initiative is to provide a Web application with simple access to native terminal features, enabling these applications to accomplish most of the actions the user has access to via native APIs. The intention is, therefore, to create a requirement document and technical specification which is not trying to compete in depth and complexity with native or java platforms, but tries and enables as many features as possible in a simple manner allowing web developers to leverage Web Applications easily and avoiding steep learning curves.

6.2 KEY ISSUE 1

Sending messages (mainly complex and large MMS and email messages) can be a time consuming task. Using synchronous methods for those actions would result in Web Applications getting blocked whenever any of these messages are in the Outbox. To avoid this issue, the usage of asynchronous methods for those particular actions is a must.

6.3 KEY ISSUE 2

Web Applications may use messaging capabilities to exchange data with other applications in other Terminals or with a different entity in the network. The Web Application may choose to silently send them and may want the sending to remain unnoticed by the user. This list of requirements addresses

this issue by allowing the developer to request message sending without leaving any trace in the platform message sent folder.

6.4 KEY ISSUE 3

Controlling which type of message is to be sent when using this interface could be an important issue for developers and users. Different technologies offer different capabilities to the developers and may have different rate implications to the end-user. For that reason, the requirements establish that the developer must have complete control over the type of message to be created and sent.

A couple of possible technical specifications for this BONDI Interface are proposed within this document annex. The first one is a simple and created from scratch specification while the second one is based in the J2ME Wireless Messaging API 1.1 [5] (JSR120).

6.5 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0150	A Web Application MUST have the possibility to select the message type (e.g. SMS, MMS or e-mail) to be used.	
IFC-0160	A Web Application MUST be able to programmatically create an SMS.	
IFC-0160.1		The Web Application MUST get access to the temporary SMS created through an object stored in its memory.
IFC-0160.2		It MUST be possible for the Web Application to provide the destination list.
IFC-0160.3		It MUST be possible for the Web Application to provide the message body.
IFC-0170	A Web Application MUST be able to programmatically send a previously created SMS (see IFC-ME-170).	
IFC-0170.1		It MUST be implemented in an asynchronous mode.
IFC-0170.2		It MUST be possible to select the function that is going to handle the result of the asynchronous operation.
IFC-0170.3		It SHOULD be possible to state whether the message must be stored within the "sent message folder" or not after the operation is completed.
IFC-0180	A Web Application MUST be able to programmatically create an MMS.	

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0180.1		The Web Application MUST get access to the temporary MMS created through an object stored in its memory.
IFC-0180.2		It MUST be possible for the Web Application to provide the destination list.
IFC-0180.3		It MUST be possible for the Web Application to provide the message subject.
IFC-0180.4		It MUST be possible for the Web Application to provide the message body.
IFC-0180.6		It MUST be possible for the Web Application to add attachments to the message.
IFC-0180.7		It MUST be possible to specify the content type (i.e. MIME type) for each attachment.
IFC-0190	A Web Application MUST be able to programmatically send a previously created MMS (see IFC-ME-190).	
IFC-0190.1		It MUST be implemented in an asynchronous mode.
IFC-0190.2		It MUST be possible to select the function that is going to handle the result of the asynchronous operation.
IFC-0190.3		It SHOULD be possible to state whether the message must be stored within the "sent message folder" or not after the operation is completed.
IFC-0200	A Web Application MUST be able to programmatically create an e-mail.	
IFC-0200.1		The Web Application MUST get access to the temporary e-mail created through an object stored in its memory.
IFC-0200.2		It MUST be possible for the Web Application to provide the "to" field.
IFC-0200.3		It MUST be possible for the Web Application to provide the "from" field.
IFC-0200.4		It MUST be possible for the Web Application to provide the "cc" field.
IFC-0200.5		It MUST be possible for the Web Application to provide the "bcc" field.
IFC-0200.6		It MUST be possible for the Web Application to provide the message subject
IFC-0200.7		It MUST be possible for the Web Application to provide the message body.
IFC-0200.8		It MUST be possible for the Web Application to add attachments to the message.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0200.9		It MUST be possible to specify the content type (i.e. MIME type) for each attachment.
IFC-0210	A Web Application MUST be able to programmatically send a previously created e-mail (see IFC-ME-210).	
IFC-0210.1		It MUST be implemented in an asynchronous mode.
IFC-0210.2		It MUST be possible to select the function that is going to handle the result of the asynchronous operation.
IFC-0210.3		It MUST be possible to specify the SMTP account ¹ to be used for sending the e-mail.
IFC-0210.5		It SHOULD be possible to state whether the message must be stored within the "sent message folder" or not after the operation is completed.

¹ An SMTP account consists of at least an SMTP server address and a port number. If omitted, the default SMTP port number will be 25. It may optionally contain a login and a password to authenticate against the SMTP server.

7 MEDIA GALLERY

This section defines the requirements for the interfaces that allow Web Applications to browse the Media Gallery, store content in the Media Gallery, (e.g. an image available in a web page through a URL) delete content from the Media Gallery or retrieve content from the Media Gallery.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0220	A Web Application MUST be able to retrieve the content list from the Media Gallery	
IFC-0220.1		A Web Application MUST be able to retrieve the number of content items in the Media Gallery.
IFC-0220.2		A Web Application MUST be able to define a set of parameters to filter the content items in the Media Gallery.
IFC-0220.3		A Web Application MUST be able to define a set of parameters to sort the content items in the Media Gallery.
IFC-0230	A Web Application MUST be able to retrieve content from the Media Gallery.	
IFC-0230.1		A Web Application MUST be able to retrieve the list of content items from the Media Gallery based on the filtering parameters and sorting options.
IFC-0230.2		A Web Application SHOULD be able to retrieve the media information of content items (eg. title, author, date, album, etc.) from the Media Gallery.
IFC-0240	A Web Application MUST be able to delete content from the Media Gallery.	

8 PERSISTENT DATA STORAGE

8.1 SCOPE

The persistent storage functional group includes all APIs that provide access to persistent storage services locally on a Terminal, including:

- File storage (e.g. serially accessed, raw binary data);
- Structured data storage (including data that is structured or typed in some way – e.g. as records, rows, tables etc. - and/or indexed in some way – e.g. in a conventional database, key/value pairs etc).

It includes data:

- That are created and maintained for exclusive use by a Widget.
- That are shared with other Widgets or applications.

8.2 KEY ISSUES

The persistent data storage APIs support a broad range of different use cases, with a correspondingly broad range of functionalities. The use cases of interest, and their relative importance, impact on the specific features required and their priority. The paragraphs below highlight the key issues and decisions needed to define an appropriate API for BONDI.

With regard to structured data, there are two key activities underway outside BONDI to provide a structured persistent data storage capability:

- The HTML5 Offline Web Applications proposal.
- Google's Gears.

These activities are now converging on a relational database model for structured data storage, accessed by a query API utilising SQL as a query language. The Google Gears implementation of this is comparatively mature. It is being used as the basis for offline operation in a number of the Google properties as well as by multiple third party libraries. The HTML5 spec is very similar to Google Gears (in respect of structured persistent data support). Multiple browser manufacturers, including Mozilla, Apple and Opera have stated that they will shortly support these features.

The momentum behind these initiatives suggests that there is no need for BONDI to stimulate the creation of a standard, and it would be disadvantageous for BONDI to create something independent and competitive.

Recommendation: it is recommended that structured data storage APIs are not developed within BONDI at this stage.

This section therefore discusses issues relating to the definition of a file system API.

8.3 TOP-LEVEL ORGANISATION OF FILE SYSTEM

It is accepted that it is not appropriate to provide an API that exposes the full file system of a terminal. The two main problems with this are:

- The file system will, in general, contain private files, and manipulation of those files could compromise:
 - the integrity of the Terminal,
 - the confidentiality and integrity of the data.
- The top-level organisation of a Terminal file system is not portably defined. It is not very useful to expect the Javascript programmer (or Terminal user) to navigate the raw Terminal file system.

Instead, all of the existing APIs or API proposals have some notion of there being a series of file system *roots*, or *mount points*, each of which is a top-level directory corresponding to a portion of the file system with a specific purpose. Examples of useful file system roots might be:

- An application's install directory;
- A user's home directory;
- A "My documents" directory (or corresponding directories for specific media types such as images or music);
- A removable storage card.

Existing APIs vary according to the precise way in which these roots are supported, and the functionality offered. Key questions are:

- **Support for well-known root names or roles.** In each of the APIs, roots have a name which is intended to be meaningful to a user – for example "CF card". The question here, however, is the degree to which those names are portably defined, or are discoverable, so that applications can be written to be portable across terminals with different file system layouts or naming conventions. The JSR75 API, for example, does not support well-known root names, but the Adobe Air API supports discovery of file system roots for `appStorageDirectory`, `appResourceDirectory`, `currentDirectory`, `documentsDirectory`, `userDirectory`, etc.

- **Support for both application-local and shared roots.** An application-local root would be a directory whose content is intended exclusively for use by the owning application. This could include application resources (i.e. the assets such as images, the application may have been deployed with) plus application storage (i.e. files containing private data created and maintained by the application). Application-local directories must be physically organised on a terminal so that they are uniquely associated with a single Widget or single Web Page. A shared root would be a directory shared between multiple applications, such as one or more directories containing Terminal Gallery data. The question here is the degree to which application-local roots and shared roots are supported. The JSR75 API, for example, does not make any statements about which roots exist – and no programmatic means to determine whether a given root is local or shared. The Adobe Air API, conversely, requires all of the portably identified roots to exist.
- **Support for dynamic roots.** Roots that correspond to removable media (i.e. a storage card) are temporary, and only exist when the removable media is in the Terminal. The question here is whether or not an API exists for an application to learn asynchronously of the creation of a root (e.g. by the insertion of a storage card). JSR75 supports dynamic roots but none of the other APIs do this.

Recommendation: it is recommended that the BONDI file system API supports:

- Roots with well-known names (or discoverable names) for at least:
 - A read-only directory containing assets derived from the application's install package (if a Widget);
 - A read/write application-local persistent directory, available both for Widgets and Web Pages;
 - A top-level root corresponding to global shared data (e.g. "My documents" or "Gallery").
- An API for discovering the default location for assets of a given MIME type, if any;
- Dynamic asynchronous indications for dynamic roots.

8.3.1 INTERACTIVE FILE/DIRECTORY PICKING

Here the question is whether or not the API provides the ability to open a "File Dialog" which allows the user to navigate the file system interactively, and pick a file or directory for an open or save operation. There are several issues relating to the provision of such an API:

- The degree to which the user is expected to understand the organisation of a file system, navigation of folders, creation of folders, etc. In APIs designed for use on a PC, it is reasonable to support this functionality because the user is generally accustomed to navigating the file system, and a precedent exists with the `<INPUT type="file">` form element. On a Terminal however, this is less likely to offer the appropriate user experience.
- The degree to which the file/directory picker mirrors the appearance and behaviour of the native file dialog for the Terminal.
- The “pop-up” risk associated with having the ability to launch the dialog programmatically. The Opera file API proposal explicitly addresses this issue by requiring implementations to open the pop-up only when the API call occurs as a direct result of real user interaction, in the way that some pop-up blockers work.
- Directory picking. The Opera API proposal includes a “BrowseForDirectory” API, but this is not implementable using the native file picker APIs on most common operating systems.

The JSR75 API and Adobe Air APIs do not include support for an interactive file picker. The Opera API proposal does include support, and such a capability is also under consideration in Google Gears.

Recommendation: it is recommended that the BONDI Interface does not include support for interactive file/directory picking.

8.4 BINARY DATA SUPPORT

JavaScript (as currently implemented in most browsers at ECMA Script version 3.x) does not provide support for efficient handling of binary data. However, a number of the use cases for file system access require the handling of binary data. The question is how this is handled in the BONDI Persistent Data Storage API.

Opera’s API and the Adobe Air API assume that there is general support for a `ByteArray` primitive. Such a primitive is defined in an ECMA Script 4 proposal, and already exists in ActionScript. The Opera API proposal also includes file read/write methods that process binary data as Strings containing Base64. Google Gears has a blob API, which is in the process of being migrated to an interface very similar to the ActionScript `ByteArray` primitive.

Recommendation: it is recommended that environments implementing the BONDI Interface also include support for ECMAScript 4 `ByteArrays`, and BONDI file system APIs assume the existence of this type.

8.5 SYNCHRONOUS VS. ASYNCHRONOUS OPERATION

The question here is whether file system operations are treated as synchronous (i.e. blocking) or asynchronous operations by the file system API. Many operations can be long-running, especially when large files such as photos are involved, pointing to a need for an asynchronous API.

Different APIs deal with this problem in different ways:

- In the Opera API proposal, most operations are blocking and synchronous but there is also the option of opening a file for asynchronous operation. In addition, there is a FileStream API which allows the serial reading of chunks of data from a file so an application can ensure that it is only blocked on file input/output for a short period of time.
- In Google Gears, operations are blocking and synchronous, but Google Gears includes a worker thread pool, which allows long-running tasks to be moved out of the main interaction thread.
- The Adobe Air API includes two versions of each API, a synchronous version and an asynchronous version.
- The JSR75 API [6] only includes blocking synchronous operations – like all other Java IO APIs, but this is expected in the Java case because it supports the creation of threads that can be used for blocking without suspending any user interaction.

The provision of support of synchronous and asynchronous operations is a general issue that affects all BONDI Interfaces, but it can be usefully informed by current practice in existing file system APIs.

Recommendation: it is recommended that BONDI includes both synchronous and asynchronous versions of all APIs that are potentially long-running.

8.6 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0250	A Web Application MUST be able to read assets delivered with the application's install package.	
IFC-0250.1		The Web Applications MUST be able to identify the root that exposes a read-only area containing the assets (if any) included in the application's install package.
IFC-0260	A Web Application MUST be able to read/write data in an exclusive area.	

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0260.1		A root MUST be available to Web Applications that expose a read/write private storage area exclusively available to that application.
IFC-0260.2		The Web Applications MUST be able to identify the root linked to the top-level directory for the exclusive area.
IFC-0260.3		The Web Application SHOULD have the possibility to request a certain amount of terminal storage in its exclusive area.
IFC-0270	A Web Application MUST be able to read/write data available in the shared file system.	
IFC-0270.1		A Web Application MUST be able to obtain a list of available root directories.
IFC-0270.2		A root MUST be available to Web Applications that expose the top-level directory containing shared files.
IFC-0270.3		A Web Application MUST be able to identify the root linked to the top-level directory for shared data.
IFC-0270.4		A function SHOULD be available to Web Applications which, given a MIME type string, identifies the default location of media assets of that type.
IFC-0270.5		A function MUST be available which permits a Web Applications to register for events corresponding to the creation or destruction of dynamic roots.
IFC-0280	The BONDI Persistent Data Storage Interface MUST provide to Web Applications an interface for interrogating and modifying roots, subdirectories within roots, and individual files.	
IFC-0280.1		For each element (roots, directories and files) within the file system, the following properties MUST at least be supported by the BONDI Persistent Data Storage Interface: <ul style="list-style-type: none"> • parent <ul style="list-style-type: none"> - permissions (type of access provided, e.g. read/write/execute), • owner, • isFile / isDirectory, • created (timestamp), • modified (timestamp), • name, • length.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0280.2		The BONDI Persistent Data Storage Interface MUST support at least the following functions (on roots, directories and files as appropriate): <ul style="list-style-type: none"> • open, • copy , • move, • createDirectory, • deleteDirectory, • deleteFile.
IFC-0290	The BONDI Persistent Data Storage Interface MUST provide to Web Applications an interface for performing read and write operations on opened files and directories.	
IFC-0290.1		The interface MUST support at least the following properties (on roots, directories and files as appropriate): <ul style="list-style-type: none"> • position, • available, • eof, • encoding.
IFC-0290.2		The interface MUST support at least the following functions (on roots, directories and files as appropriate): <ul style="list-style-type: none"> • close, • read, • readLine, • readBytes, • readBase64, • write, • writeLine, • writeBytes, • writeBase64.

9 DEVICE STATUS AND PROPERTIES

9.1 SCOPE

The aim of this section is to provide a single location for all the information related to the status and properties of the Terminal that influence the life time of the web application and its internal algorithms. Those status and properties may be read and some of them may be set programmatically.

It may be useful to regard the persistent feature as the property (always set to the same value) whereas the dynamic feature as those that may change – from the application point of view i.e., status. Additionally we must distinguish between read-only and programmatically modifiable statuses.

9.2 ISSUE 1

As the access to all the properties and status is supposed to be done uniformly, this section includes the status and properties related to other interfaces described in other chapters, like Camera or API availability.

9.3 ISSUE 2

There are many other initiatives (such as OMA DPE [7]) in which vocabularies and descriptions for some of the properties discussed in the document are included. Where applicable, this document references these documents. For the sake of consistency, these external specifications are referred to where ever possible.

9.4 ISSUE 3

The number of properties is likely to evolve with the time and the addition of new features to the WREs. In order to provide a flexible solution allowing future extensions, a decoupling between the APIs and the Data Models is recommended.

9.5 ISSUE 4

It must be clearly specified what “always” means in the context of persistent features if such a distinction is to be introduced. Installable WREs make virtually all features dynamic. Therefore, there could be another “meta” level of the properties that would introduce another property, namely, dynamic (i.e. becomes a status in the above terminology).

9.6 ISSUE 5

The notifications of the status being changed may be provided to the application based on the mechanisms described in Section 15, System Events or such a mechanism must be defined in this section.

9.7 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300	A Web Application MUST be able to obtain information about system properties	

9.7.1 BATTERY

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.1		It MUST be possible to obtain information on whether the battery is being charged or not (see BatteryBeingCharged in OMA DPE [7]).
IFC-0300.2		It MUST be possible to obtain the battery's remaining power (see BatteryStatus in OMA DPE [7]).

9.7.2 CONNECTIVITY

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.3		It MUST be possible to obtain information on which a Terminal's wireless connection functions are available or not (i.e. the hardware is on or off).
IFC-0300.4		It MUST be possible to obtain information about the types of wireless connection functions that are supported (i.e. the required hardware is present) by a Terminal, e.g. cellular, WLAN, Bluetooth.
IFC-0300.4B		It MUST be possible to obtain information about the types of wireless connection functions that are connected e.g. cellular, WLAN, Bluetooth.
IFC-0300.5		It MUST be possible to obtain the signal strength of a Terminal's wireless connection function.

9.7.3 USER INTERACTION SETTINGS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.6		It MUST be possible to obtain information on whether the sound device in the Terminal is muted or not.
IFC-0300.7		It MUST be possible to obtain the volume setting of the Terminal.
IFC-0300.7A		It MUST be possible for Web Applications to retrieve the current settings of the sound mode set in the active user profile.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.7B		It MUST be possible for Web Applications to retrieve the range of possible beeping duration and frequencies.
IFC-0300.7C		It MUST be possible for Web Applications to retrieve the range of possible vibration duration times and intensities.
IFC-0300.7D		It MUST be possible for Web Applications to retrieve the range of possible light on duration and intensities.
IFC-0300.7E		It MUST be possible for Web Applications to retrieve the possible targets to the switch on light action (e.g. main screen, keypad etc.).
IFC-0300.7F		It MUST be possible for Web Applications to determine if the Terminal supports landscape and portrait screen orientations.

9.7.4 MEMORY

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.8		It MUST be possible to obtain the total internal memory size of the Terminal (see TotalInternalMemorySize in OMA DPE [7]).
IFC-0300.8A		It MUST be possible to obtain the available internal memory size of the Terminal (see AvailableInternalMemorySize in OMA DPE [7]).
IFC-0300.8B		If a removable data storage card is inserted on the Terminal it MUST be possible to obtain the total memory size of the data card (see TotalCardMemorySize in OMA DPE [7]).
IFC-0300.8C		If a removable data storage card is inserted on the Terminal it MUST be possible to obtain the available memory size of the data card (see AvailableCardMemorySize in OMA DPE [7]).

9.7.5 DEVICE

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9A		It MUST be possible to obtain the Terminal's vendor name, model name and version number.

9.7.6 OPERATING SYSTEM

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9B		It MUST be possible to obtain the Operating System vendor name, product name and version number for the available Operating System.

9.7.7 WEB RUNTIME ENVIRONMENT

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9C		It MUST be possible to obtain the Web Runtime Environment's vendor name, product name and version number.

9.7.8 WEB BROWSER

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9D		It MUST be possible to obtain the Web Browser's vendor name, product name and version number, for the available and default Browser.

9.7.9 JAVA RUNTIME ENVIRONMENT

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9E		It MUST be possible to obtain the Java Runtime Environment (JRE) vendor name, product name and version number for the available and default JRE's.

9.7.10 OMA PUSH CLIENT

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9F		It MUST be possible to obtain the OMA Push Client vendor name, product name and version number for the default Push Client.

9.7.11 OMA MULTIMEDIA MESSAGING CLIENT

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.9G		It MUST be possible to obtain the OMA Multimedia Messaging (MMS) Client vendor name, product name and version number for the default MMS Client.

9.7.12 CERTIFICATES

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.10		It MUST be possible to obtain the list of certificates that the Web Runtime Environment supports to verify the signatures of Widget Resources.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.11		It MUST be possible to obtain the list of certificates that the Web Runtime Environment supports to verify a server's identity for SSL/TLS sessions.
IFC-0300.12		It SHOULD be possible to obtain the list of certificates that the Terminal and all its Application Execution Environments (AEE) support.

9.7.13 CAMERA

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.13		It MUST be possible to identify whether the Terminal has its primary camera enabled (see PrimaryCameraEnabled in OMA DPE [7]).
IFC-0300.14		It MUST be possible to identify whether the Terminal supports (at least) one camera (see PrimaryCameraPresent in OMA DPE [7]).
IFC-0300.15		It MUST be possible to identify whether the Terminal has its secondary camera enabled (see SecondaryCameraEnabled in OMA DPE [7]).
IFC-0300.16		It MUST be possible to identify if the Terminal supports a second camera (see SecondaryCameraPresent in OMA DPE [7]).
IFC-0300.17		It MUST be possible to query the current resolution of the primary camera (see PrimaryCameraResolution in OMA DPE [7]).
IFC-0300.18		It MUST be possible to query the current resolution of the secondary camera (see SecondaryCameraResolution in OMA DPE [7]).
IFC-0300.19		It MUST be possible to identify whether the primary camera supports zoom functionality.

9.7.14 INPUT/OUTPUT

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0300.20		It MUST be possible to obtain the list of input devices supported by the Terminal (e.g. keypad, touch screen, stylus, keyboard etc.).
IFC-0300.21		It MUST be possible to obtain the Terminal audio input encoders (see AudioInputEncoder in OMA DPE [7]).
IFC-0300.22		It MUST be possible to obtain the Terminal input character set (see InputCharset in OMA DPE [7]).
IFC-0300.23		It MUST be possible to obtain the Terminal output modality (see OutputModality in OMA DPE [7]).
IFC-0300.24		It MUST be possible to identify whether the Terminal is actuating "tethered" to another Terminal, giving the latter some kind of wireless connection (see Tethering in OMA DPE [7]).

10 COMMUNICATION LOG

10.1 SCOPE

The interfaces within this category should provide application developers with access to information on recent calls (missed, received and initiated) and messages (sent, received, drafts and templates) that are maintained by the respective native applications. The Communication Log (CL) functionality is not intended to provide access to the call handler or messaging services directly but can be used to list the information that have been stored in the CL.

Information in the CL MUST be possible to reuse e.g. in order to initiate a call or read a message using the related functional interface (subject to relevant access control policies). A detailed list of all the possible fields is available in the vCard specification developed by the IMC (Internet Mail Consortium).

The following high level requirements (or use cases) will be supported by the Communication Log;

- Retrieve Call Log information.
- Retrieve Message Log Information.
- Manage Information Elements for Call and Message Log entries.

10.2 KEY ISSUE 1 - EXTENSIBILITY

Existing Terminals support a range of communication technologies having different sets of attributes related to their basic call and messaging services. A voice call log entry typically includes information such as originating call number (if not blocked) but may also provide additional information such as media specific extensions (voice call or video call etc). For a messaging log the extent of logged information that can be returned is even more extensive, for example, comparing a simple SMS with an e-mail that may include extensive information such as message headers.

Recommendation 1

The Communication Log Interface should facilitate retrieval of an arbitrary number of information elements (subject to relevant policy and privacy settings) relevant to the used media e.g. SMS, e-mail, Voice Call, Video Call and so on.

10.3 KEY ISSUE 2 - SECURITY AND PRIVACY CONSIDERATIONS

The Communication Log is potentially a very sensitive resource for an end-user and necessary precautions need to be taken how to manage the access to the CL information on the BONDI Security Framework. The potential for

abuse is very high as many social network services attempt to leverage harvesting of contacts (which is not necessarily done in a rogue fashion).

Recommendation: Access to the Communication Log should be regarded as sensitive as any access to the contacts and PIM database of a Terminal. It is potentially even more sensitive as the CL also includes information when a call was made that can then be used to track user behaviour.

10.4 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0310	A Web Application MUST be able to retrieve information from the recent call lists that are maintained by the respective native telephony application.	
IFC-0310.1		It MUST be possible to retrieve the information from the missed calls list.
IFC-0310.2		It MUST be possible to retrieve the information from the received calls list.
IFC-0310.3		It MUST be possible to retrieve the information from the initiated calls list.
IFC-0310.4		It MUST be possible to indicate a selection criteria (or folder) for the call list whose information shall be retrieved, i.e. Received, Missed or Initiated.
IFC-0320	A Web Application MUST be able to retrieve information from the message lists that are maintained by the respective native applications.	
IFC-0320.1		It MUST be possible to retrieve the information from the sent message lists.
IFC-0320.2		It MUST be possible to retrieve the information from the received message lists.
IFC-0320.3		It MUST be possible to retrieve the information from the draft message lists.
IFC-0320.5		It MUST be possible to specify a selection criteria (or folder) for the message lists whose information shall be retrieved, i.e. Sent, Received or Draft.
IFC-0320.6		It MUST be possible to differentiate the message type such as SMS, MMS and e-mail between the various message lists.
IFC-0320.8		It MUST be possible to specify which information elements to retrieve from the Communication Log such as TimeStamp, PhoneNumber, destination e-mail address, subject, etc.

11 PERSONAL INFORMATION

11.1 SCOPE

Many of the use cases listed in Section 3 highlight the need for Web Applications to have access to personal information data. Personal information is defined as information included in the contact lists, calendar database and task lists.

This API is expected to support the following data type and use cases:

- Management of Information Elements.
- Management of PIM Database entries.
- Search for events in the PIM Database.

11.2 KEY ISSUE 1: MANAGING DATA FROM DIFFERENT SOURCES

Personal Information is not only restricted to data stored on the Terminal but applies also to other locations such as SIM/UICC or memory cards. There are different approaches when dealing with data coming from different lists:

- Manage each list separately.
- Combine all the data from the different sources into a unique.
- Allow developers to select their preferred option.

In order to simplify the application development, this API should abstract the source of data and combine all the data whenever possible. Additionally, the API should provide sufficient tools to enable application developers to identify the list in which the data is physically stored.

When a Web Application requests the complete list of any given type of data (e.g. contacts) all the entries (e.g. in the UICC and in the Terminal memory) should be returned. However, there should also be a parameter on each of the entries indicating the list in which the data is stored.

It is also recommended, that whenever a search operation is performed, all lists should be scanned for results.

11.3 KEY ISSUE 2: FIELD SUPPORT

Each personal information element may support different fields (e.g. the fields defined in vCard, vCalendar, vTodo or a subset of them), providing as much information as possible about a database entry.

Usually, different lists do not have the same fields supported, as the subset chosen by the manufacturers is different.

This document recommends a minimum set of fields (the subset of those defined in vCard, vCalendar and vTodo) that the WRE should support for each PIM List. These subsets provide the relevant information about a PIM entry without compromising platform portability.

It is acknowledged that some PIM databases may not have all the information required (e.g. contact information stored in SIM/UICC tends not to include all the fields defined). In case a database does not support all the mandated fields, default values should be added to those database entries in the returned lists.

In order to allow developers to use additional fields from those defined as mandatory, it is also recommended to offer the following mechanisms in order to:

- Allow Web Applications to retrieve all the supported fields for a given list
- Check if a field name is supported in a list.

11.4 EVENT MANAGEMENT

This set of requirements defines the minimum functionality that the WRE must offer with regards to the management of events in the PIM database (or Calendar). An event may have different fields depending on the support of the native event databases. A detailed list of all the possible fields is available in the iCalendar specification [8].

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0330	A Web Application MUST be able to retrieve information about the supported information elements/fields.	
IFC-0330.1		It MUST be possible to retrieve a list of supported named information elements such as Start, End, Title, Reminder and Location etc.
IFC-0330.2		It MUST be possible to retrieve the constraint on storage size for each supported information element.
IFC-0340	A Web Application MUST be able to manage the events stored in the PIM database.	
IFC-0340.1		Web Applications MUST be able to retrieve all events with all its named information elements.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0340.4		Web Applications MUST be able to modify an event in the PIM database.
IFC-0340.5		It MUST be possible to add events to the PIM database.
IFC-0340.6		Web Applications MUST be able to remove an event from the PIM database.
IFC-0340.7		Web Applications MUST be able to delete all the events from the PIM database.
IFC-0340.8		For each event, the following information MUST be accessible: <ul style="list-style-type: none"> • Location, • Note, • Summary, • Identifier, • Start, • End, • Alarm.
IFC-0340.9		For each event it MUST be possible to set a value to any named (or otherwise identified) information element such as Start, End, Title, Reminder, Location etc.
IFC-0350	A Web Application MUST be able to search for events in the PIM database.	
IFC-0350.1		Web Applications MUST be able to retrieve a list of all events with their named information that match character patterns in a specified set of named information elements.

11.5 CONTACT MANAGEMENT

This set of requirements defines the minimum functionality that the WRE must offer with regards to the management of contact lists in the PIM database. A contact may have different fields depending on the support of the native contact databases. A detailed list of possible fields is available in the vCard specification [9] developed by the Internet Mail Consortium (IMC).

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0360	A Web Application MUST be able to retrieve information about the supported information elements/fields.	
IFC-0360.1		It MUST be possible to retrieve a list of supported named information elements such as Name, Address, Phone Number etc.
IFC-0360.2		It MUST be possible to retrieve the constraint on storage size for each supported information element.
IFC-0370	A Web Application MUST be able to manage the PIM database contacts.	

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0370.1		Web Applications MUST be able to access the different contact lists regardless of their storage area (e.g. SIM/UICC, Terminal and Memory Card).
IFC-0370.2		Web Applications MUST be able to retrieve all the contacts with all named information elements stored in the PIM Database.
IFC-0370.4		Web Applications MUST be able to modify a contact in the PIM Database.
IFC-0370.4A		Web Applications MUST be able to add contacts to the PIM database.
IFC-0370.6		Web Applications MUST be able to remove a contact from the PIM Database.
IFC-0370.7		Web Applications MUST be able to remove all the contacts from the PIM Database.
IFC-0370.8		For each contact, the following information MUST be accessible to Web Applications: <ul style="list-style-type: none"> • Name, • Nickname, • Address, • Identifier, • Photo, • Telephone, • Email.
IFC-0370.9		For each contact it MUST be possible to set a value to any named (or otherwise identified) information element such as Name, Address, Telephone etc.
IFC-0380	A Web Application MUST be able to search for contacts in the PIM database	
IFC-0380.1		Web Applications MUST be able to retrieve a list of all contacts with their named information that match character patterns in a specified set of named information elements.

11.6 TASK MANAGEMENT

This set of requirements defines the minimum functionality that the Web Runtime must offer with regards to Task Management in the PIM database. A task may have different fields depending on the support of the native databases. A detailed list of possible fields is available in the vTodo description part of the iCalendar specification [8].

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0390	A Web Application MUST be able to retrieve information about the supported information elements/fields.	

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0390.1		It MUST be possible to retrieve a list of supported named information elements such as Priority, Summary etc.
IFC-0390.2		It MUST be possible to retrieve the constraint on storage size for each supported information element.
IFC-0400	A Web Application MUST be able to manage the PIM Database tasks.	
IFC-0400.1		Web Applications MUST be able to retrieve all the tasks stored in the PIM Database with all its named information elements.
IFC-0400.3		Web Applications MUST be able to modify a task in the PIM Database.
IFC-0400.3A		Web Applications MUST be able to add tasks to the PIM Database.
IFC-0400.5		Web Applications MUST be able to remove a task from the PIM Database.
IFC-0400.6		Web Applications MUST be able to delete all the tasks from the PIM Database.
IFC-0400.7		For each task the following information MUST be accessible to Web Applications: Priority, Note, Summary, Identifier, Due, Completed.
IFC-0400.8		For each task it MUST be possible to set a value to any named (or otherwise identified) information element such as Priority, Note, Summary etc.
IFC-0410	A Web Application MUST be able to search for tasks in the PIM database.	
IFC-0410.1		Web Applications MUST be able to retrieve a list of all tasks with their named information that match character patterns in a specified set of named information elements.

12 LOCATION

12.1 SCOPE

The BONDI Location Interface allows a Web Application to retrieve the current or last location of the Terminal. This can be as a standalone event (a one-shot request/response) or as a series of ongoing notifications (based on a listening interface in the Web Application).

This interface should provide the capability to retrieve the current Terminal location from the most suitable Location Provider² using the most suitable Location Method³ supported by the Terminal. This requires the BONDI Location Interface to be Location Technology agnostic (i.e. not reliant on a single location resolution technology such as GPS).

Where the current Terminal location is unavailable (e.g. network unavailable), the last known Terminal location information should be provided as stored on the Terminal.

12.2 KEY ISSUES

W3C are currently in the process of defining GeoLocation API specifications. The Java Community Process (JCP) has also recently published a Location API (JSR-179) for use in Java-based environments, so a suitable alignment between the final BONDI Location Interface and these external initiatives must be decided and agreed.

Recommendation: Care must be taken to ensure that the final BONDI Location API is compliant with the W3C GeoLocation API specifications.

At a further level, care should be taken to re-use the concepts and structure developed in the Java Location API (JSR-179) to the greatest extent possible.

12.3 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0420	A Web Application SHOULD be able to manage and select the Location Providers available.	
IFC-0420.1		A Web Application SHOULD be able to obtain a list of Location Providers available.

² Location Provider is a programmatic representation of an individual hardware component (e.g. GPS Component)

³ Location Method is a programmatic representation of the Location Resolution Procedure (e.g. Assisted GPS) used to obtain the location data

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0420.2		For each Location Provider, the Location Method used by this Location Provider MUST be specified (as part of the returned information).
IFC-0420.3		A Web Application SHOULD be able to select a preferred Location Provider.
IFC-0420.4		At least one Location Method per Location Provider SHOULD be supported.
IFC-0420.5		Location Methods MAY include: <ul style="list-style-type: none"> • Angle of Arrival Method, • Assisted Method, • Cell-Id Method, • Network Based Method, • Terminal Based Method, • Time Difference Method, • Time of Arrival Method, • Unassisted Method, • WiFi Neighbourhood Method, (See JSR 179 [10] and [11] for further details).
IFC-0430	A Web Application MUST be able to obtain a one-shot location update.	
IFC-0430.1		If the one-shot location update request fails, the Web Application SHOULD be returned to the last known location retrieved by the Terminal or empty.
IFC-0430.2		The one-shot location update response object MUST include: <ul style="list-style-type: none"> • Latitude (-90 to +90 WGS84 degrees), • Longitude (-90 to +90 WGS84 degrees), • Altitude, • Horizontal Accuracy (0 to 3.4028e+38 metres), • Vertical Accuracy, • Timestamp (YYYY-MM-DDThh:mm:ssZ), • Heading, • Velocity
IFC-0440	A Web Application MUST be able to retrieve the last known location update information when the current location information cannot be determined on the Terminal.	
IFC-0450	A Web Application MUST be able to obtain periodic location updates.	
IFC-0450.1		The Web Application MUST be able to implement and specify a location listener or an equivalent method of asynchronously receiving location updates ⁴ .
IFC-0450.2		A Web Application MUST be able to stop periodic location updates. On completion, the associated location listener interface MUST NOT receive any further location updates.

⁴ “Listener” can be a loaded term. Platform asynchronous operation can also be extended to the Javascript layer without using the Javascript event listener.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0450.3		The Web Application MAY be able to specify a distance filter for the control of location notification intervals. The distance filter parameter states the minimum distance the user must move before an update event is generated and sent to the Web Application.
IFC-0460	The Web Application SHOULD be able to programmatically specify a timeout value for any location information request.	
IFC-0460.1		If a location request timeout (on either the one-shot method [IFC-LOC-440] or the periodic location update method [IFC-LOC-460]) is reached, the request SHOULD return the last known location information.
IFC-0460.2		If last known location update information is unavailable, the request SHOULD return an empty value or an exception.
IFC-0470	The Web Application MUST be able to programmatically specify the desired accuracy for any location information request.	
IFC-0470.1		The Web Application MUST be able to include in any location request the desired accuracy
IFC-0480	A Web Application MUST be able to query the orientation of a Terminal.	
IFC-0480.1		The orientation response object SHOULD include: <ul style="list-style-type: none"> • azimuth - the Terminal's horizontal compass bearing in relation to true north: 0 to 360 degrees.
IFC-0480.2		The orientation response object MAY include <ul style="list-style-type: none"> • pitch: the Terminal's angle in a vertical plane orthogonal to the ground: 90 to 90 degrees), • roll: the Terminal's rotation in degrees around its own longitudinal axis: 180 to 180 degrees).

13 USER INTERACTION

13.1 SCOPE

This group includes all APIs that provide mechanisms to interact with the end-user or configure the way in which the user visualizes or navigates Web Applications. This category includes features such as:

- Navigation mechanisms, e.g. tabbed or cursor style.
- Soft-key and menu configuration.
- Effectors, e.g. vibration, sounds and screen brightness.
- Configuration of general appearance, e.g. portrait/landscape orientation, use of full screen and move to background or foreground.
- Behaviour configuration when moving from foreground to the background or vice versa.

13.2 KEY ISSUE 1

For Terminals not supporting a touch-screen, there are usually two different ways to navigate within a web page:

- Cursor navigation: Similar to the mouse navigation in a PC. There is a pointer controlled by the user that allows navigation of the entire screen.
- Tabbed navigation: The user can move the focus between different components on the page using the keys, but there is no cursor that allows him/her to navigation the components of the Widget.

Recommendation: In Terminals without touch-screen, it is recommended that Web Applications are able to toggle between different navigation modes.

13.3 KEY ISSUE 2

Terminals have small screens compared to a desktop computer therefore it is not as easy or convenient to include a large number of elements, buttons or control tools. In order to minimize the amount of screen area used to provide the management of commands, it is essential Web Applications support mechanisms to create and manage menus that appear only when requested by the user.

Most Terminal user interfaces offer the ability to interact with the user using soft-keys. Soft-keys may be mapped to functions or to a menu pop-up. Users are familiar with this interaction mechanism, and hence, it is a good idea to allow Web Applications to use the same interaction model.

Conversely, the user experience of the Terminal should not be broken, for example if a soft-key is always used for a given purpose and Web Applications are prohibited from modify it.

Recommendation: Terminals supporting soft-key UIs should allow Web Applications to use soft-keys to interact with the user without breaking the general Terminal user experience.

13.4 KEY ISSUE 3

In order to attract user attention in some situations (e.g. an incoming event or an error), it is desirable to support the possibility of interaction with the user using mechanisms other than the Web Application graphical interface.

Examples of mechanisms that may be used for that purpose are:

- Turn the Terminal vibration ON for a specific duration.
- Make the Terminal beep.
- Turn on or vary light intensity for the screen, keypad, etc.

It should also be highlighted that general user preferences should not be overridden by Web Applications (e.g. the Terminal should not beep while in silent mode or should not vibrate if vibration is deactivated).

Recommendation: Terminal functionalities that allow the interaction with the user should be available for Web Applications.

13.5 KEY ISSUE 4

Many Terminals offer the possibility to modify the display orientation (i.e. portrait or landscape). Some Web Applications may be designed for optimal use in a particular orientation. Hence, it may be helpful for the API to have the possibility to request the display orientation.

Additionally, in order to utilise the spare Terminal screen space, it should be possible to use the entire screen, without displaying elements like the soft-key labels or the status bars.

Recommendation: Web Applications should have the possibility to choose between the supported orientation modes and the use of the entire Terminal screen.

13.6 KEY ISSUE 5

Some Terminals offer the capability to execute several applications virtually and concurrently, allowing one to be in the foreground, while the rest operate in the background. When an application is moved from the background to the

foreground or vice versa, it may be required to perform a set of actions, e.g. change the user status in an instant messaging client. Therefore, it is important for Web Applications to be notified when its foreground/background execution status changes.

Additionally, there are some scenarios in which Web Applications should have the possibility to move from background to foreground or vice versa. For instance, it may be useful for a messaging Web Applications running in the background to request to be moved to the foreground when an incoming message is received.

Recommendation: Web Applications should have the possibility to request that the user move them from background to foreground or vice versa. If it is the end user requesting that the execution status is changed then the Web Application should be notified about the change.

13.7 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0490	A Web Application MUST be able to select the navigation mode to be used.	
IFC-0490.1		It MUST be possible for Web Applications to select the cursor navigation to be used.
IFC-0490.2		It MUST be possible for Web Applications to select tabbed navigation to be used.
IFC-0500	A Web Applications SHOULD be able to modify the menus and actions linked to its own soft-keys.	
IFC-0500.1		It SHOULD be possible for Web Applications to add new elements or sub-menus to the menu to be displayed when pressing a soft-key.
IFC-0500.2		It SHOULD be possible to configure the Javascript method invoked when selecting a menu option and a soft-key.
IFC-0500.3		It SHOULD be possible to configure the text label of a menu option and a soft-key.
IFC-0500.4		It SHOULD be possible for Web Applications to remove elements from menus, sub-menus and soft-keys.
IFC-0500.5		It SHOULD be possible for Web Application to retrieve the list of menu items within a menu and sub-menu.
IFC-0500.6		It SHOULD be possible for Web Applications to receive notification when a soft-key has been pressed.
IFC-0510	A Web Application MUST be able to activate the Terminal vibrator.	
IFC-0510.1		It MUST be possible for Web Applications to turn Terminal vibration on for a specified duration with the specified intensity.
IFC-0510.4		It MUST be possible for Web Applications to request that Terminal vibration is stopped.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0520	A Web Application MUST be able to make the Terminal beep.	
IFC-0520.1		It MUST be possible for Web Applications to request that the Terminal uses a beep tone of a specified duration and frequency.
IFC-0530	A Web Application MUST be able to switch on and off the light of different parts of the Terminal.	
IFC-0530.1		It MUST be possible for Web Applications to request a specific target to switch its light on for a specified duration with the brightness defined by a specified light intensity.
IFC-0530.2		It MUST be possible for Web Applications to request a specific target to switch its light off for a specified duration. After the switch-off duration, the default light status determined by the Terminal MUST apply.
IFC-0540	A Web Application MUST be able to select the Terminal orientation to be used.	
IFC-0540.1		It MUST be possible for Web Applications to change the orientation of a widget's screen to the landscape mode.
IFC-0540.2		It MUST be possible for Web Applications to change the orientation of a Web Applications screen to the portrait mode.
IFC-0550	A Web Application MUST be able to manage the general elements that are visible in the Terminal screen within its display space.	
IFC-0550.1		It MUST be possible for Web Applications to request to hide the soft-keys.
IFC-0550.2		It MUST be possible for Web Applications to request to show the soft-keys.
IFC-0550.3		It MUST be possible for Web Applications to request to hide the Terminal status bar.
IFC-0550.4		It MUST be possible for Web Applications to request to show the Terminal status bar.
IFC-0560	A Web Application MUST be able to register for events related with changes in its execution status (i.e. foreground or background).	
IFC-0560.1		It MUST be possible for Web Applications to assign a call-back function for being invoked when the Web Application gains focus (i.e. is moved from background to foreground).
IFC-0560.2		It MUST be possible for Web Applications to assign a call-back function for being invoked when the Web Application loses focus (i.e. is moved from foreground to background).
IFC-0570	A Web Application MUST be able to request a change to its execution status (i.e. foreground or background).	
IFC-UI-570.1		It MUST be possible for Web Applications to request execution be moved from the foreground to the background.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-UI-570.2		It MUST be possible for Web Applications to request execution be moved from the background to the foreground.

14 CAMERA

14.1 SCOPE

The use of the camera API is not confined to taking pictures, videos or storing the resulting files in a local folder. The API may also be used as a necessary tool for accomplishing specific tasks in applications and services. Examples of these are:

- Use of the handset as 2D barcode reader e.g. for tourist or cultural services.
- Sharing photos in a social networking service e.g. during special events one can upload pictures and have them immediately visible on the big screen on the stage.
- Opening a video streaming during an instant messenger communication.

So the minimum camera functionalities required by the use cases are the capability to take a snapshot or a video, save the resulting object as a file and make it available to the calling web application. But for more interactive applications, complete control of the camera becomes a necessity.

14.2 ISSUE 1

Terminal camera characteristics and features differ by Terminal manufacturer and model. Moreover, different cameras may have different settings for the same feature. This results in interworking and camera control problems when the web application is run across multiple platforms and Terminals.

Recommendation: It is recommended that applications can gather a complete set of information on the camera controllable features (which features are controllable and which values they can assume). Additionally, a correct sequence of operations is to be defined in order to reach the same expected result on different cameras.

14.3 ISSUE 2

Passing camera control ability to web applications raises a security risk in, but not limited to, the privacy sphere.

Recommendation: It is recommended that applications inform the Terminal user that the camera is activated. Notwithstanding this recommendation it is further recommended that a deeper analysis and discussion on this issue is required.

14.4 REQUIREMENTS

Two methods of controlling the camera are defined:

- Launching the native application: all camera application features are supported that are exposed via native APIs to other applications but there is little integration with the Web Application
- Using the BONDI Camera Interface: greater integration with the Web Application however reduced access to the native camera features.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0580	A Web Application MUST be able to embed the camera viewfinder and allow control of the camera settings.	
IFC-0580.1		It MUST be possible to specify the camera to use (e.g. rear or front).
IFC-0580.2		It MUST be possible to embed the camera viewfinder.
IFC-0580.3		It MUST be possible to retrieve the picture immediately after being taken or the video immediately after recording stops.
IFC-0580.4		It MUST be possible to set the resolution.
IFC-0580.5		It MUST be possible to set the zoom level.
IFC-0580.6		It MUST be possible to set the brightness.
IFC-0580.7		It MUST be possible to set the white balance mode.
IFC-0580.8		It MUST be possible to set the contrast.
IFC-0580.9		It MUST be possible to set night mode.
IFC-0580.10		It MUST be possible to control the focus mode (if more than one).
IFC-0580.11		It MUST be possible to set the focus if focus mode is manual.
IFC-0580.12		It MUST be possible to launch a picture shot.
IFC-0580.13		It MUST be possible to set the quality of the picture.
IFC-0580.14		It MUST be possible to set the flash mode (if flash is present) and light mode (if light is present).
IFC-0580.15		It MUST be possible to set the sequence mode (single or multi shot).

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0580.16		It MUST be possible to set the self-timer.
IFC-0580.17		It MUST be possible to launch a video recorder start.
IFC-0580.18		It MUST be possible to launch a video recorder stop.
IFC-0580.19		It MUST be possible to set the maximum length of the video.
IFC-0580.20		It MUST be possible to set the video frame rate.
IFC-0580.21		It MUST be possible to disable audio for video recording or stream.

15 SYSTEM EVENTS

The BONDI System Events Interface is intended to enable JavaScript-based entities to subscribe to meaningful global events on a Terminal in a consistent manner. System events are viewed here as separate and different from input events, which are expected to be handled by the UI framework.

A basic structure for events is proposed, taking into account input from numerous specifications and implementations currently in use, including, Java JMS, LiMo Foundation Event Delivery etc. The events structure features into three components:

1. A hierarchical **event topic**, which may look something like “bondi.push” or “bondi.application.state”. Partial topics (topic prefixes) such as “bondi.application.*” are expected to be meaningful in subscriptions to classes of events.
2. An optional set of named **properties**, consisting of specific meaningful names depending on topics, which can be used in filtering. Properties may or may not be present in particular events. If, for example, the “bondi.sms” event supports port numbers, a property by the name of “port” can be used in filtering expressions associated with subscriptions, such as “port > 2000”.
3. An optional and opaque event **body**, its structure depending on particular topic.

15.1 REQUIREMENTS

An initial list of to-be-defined topics is proposed, this list is not considered to be definitive or complete.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0590	A Web Application MUST be able to subscribe to system events in a uniform manner, independent of the nature of the original event.	
IFC-0590.1		A common high-level event structure MUST be defined.
IFC-0590.2		The event structure MUST include a hierarchical event topic, event properties upon which events can be filtered and an opaque event body, with structure specific to a particular event topic.
IFC-0590.3		Subscription based on event topics MUST be supported.
IFC-0590.4		Subscription based on partial topic specification by using a small set of wildcards MUST be supported.

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-0590.5		Simultaneous subscription to multiple events MUST be supported.
IFC-0590.7		Event Publishing functionality MUST be supported.
IFC-0590.9		Requests for exclusive subscription to events MUST be supported.
IFC-0600	If the Web Application start-up is based on an event that is supported by the underlying platform, the API MUST be provided to grant the Web Application access to the initiating event.	
IFC-600.1		A function MUST be provided to determine whether the Web Application was started due to an event.
IFC-600.2		A function MUST be provided to retrieve the object describing the initiating event.
IFC-610	A comprehensive set of event topics with associated properties and body semantics MUST be supported if the underlying platform allows for it.	
IFC-610.1		An event object MUST be defined reflecting application status change, e.g. start/stop and install/uninstall.
IFC-610.2		An event object MUST be defined reflecting Terminal use mode change, e.g. flip opened/closed or slide opened/closed.
IFC-610.3		An event object MUST be defined reflecting SIM/UICC changes, e.g. new SIM/UICC, whether during the boot or changed on the fly, SIM/UICC file change etc.
IFC-610.4		An event object MUST be defined reflecting change in removable storage status.
IFC-610.5		An event object MUST be defined reflecting SMS message arrival.
IFC-610.6		An event object MUST be defined reflecting application-directed message arrival. Examples of application-directed messaging services are WAP Push or SIP Push.
IFC-610.7		An event object MUST be defined reflecting voice call related events, e.g. incoming call, call ended etc.
IFC-620	Web Applications MUST be able to subscribe to be notified whenever a device property changes	
IFC-620.1		Web Applications SHOULD be able to specify a threshold or the granularity upon which they should be notified on changes
IFC-620.2		Web Applications MUST be able to subscribe to events related with any property exposed in the Device Status Interface.

16 APPLICATION SETTINGS

16.1 SCOPE

The application settings interface includes all APIs and facilities that relate to application settings and preferences, including (if supported):

- Static parameters defined by the author of a Web Application;
- Parameters configurable by the author or publisher of a Web Application to repurpose it to a specific Terminal, locale etc., at design time, or at purchase/provisioning time;
- Parameters read and/or modified programmatically by the Web Application when it runs.

The scope does not include metadata required solely to support a provisioning mechanism, or other installation-related metadata, e.g. package size, origin, user data quota, declared properties relating to security policy etc.

There is a partial overlap of this functional group with the “persistent data storage” group, in that each could be used by an application to store or modify parameters persistently; in particular, a simple writable “properties” API could be very similar to the original WHATWG/HTML5 structured persistent data facility based on key/value pairs. However, the emphasis in this section is on APIs to support:

- Application settings and configuration parameters (as opposed to application data generally);
- Very simple and limited support for structured data rather than compound data types and/or relationships.

Specific formats, like the use of an XML representation to communicate settings as part of a provisioning protocol, are out of scope of this section, but the requirements here obviously impact on the structure and content of the data that any such format would need to support.

16.2 KEY ISSUE 1

Relationship to device management: It is possible to imagine a properties API that provides access not only to a set of properties defined locally for an application, but to a global tree of properties, including those managed by a device management system. The primary motivating use-case for a device management API might be the creation of various “settings” or “profiles” for Web Applications.

There is clearly a potential overlap between the APIs, and a settings API that could potentially be generalised to meet the wider device management need. However, in a fully-featured device management properties API, all of the well-known questions arise relating to privacy and access control, and management and portability of the properties namespace.

Recommendation: it is recommended that the settings API is confined to supporting local application properties and does not attempt to expose a global properties tree. Further consideration should be given to uses cases, priority and definition of a device management API.

16.3 KEY ISSUE 2

Writing property: The question here is whether a settings API should support writing of properties or just enable reading of pre-set properties. For example, in addition to a “getAppProperty” API there could be a “setAppProperty” API.

The arguments in favour of supporting read/write capability are:

- It is a natural and symmetrical extension of a property read API;
- It would provide a simple and lightweight way for applications to manage a limited number of settings or preferences;
- It parallels the practice in similar APIs such as the Windows registry.

The arguments against supporting read/write capability are:

- It is unnecessary given that the same effect could be achieved using other means, such as the persistent data storage API and a possible data management API;
- It introduces a number of complications to the API and the associated implementation, such as:
 - the need to specify which properties are writable and which are not;
 - the need to determine the lifecycle of written properties, whether or not they are retained when an application is upgraded, for example;
 - the need to support a quota for the maximum number of properties or size of data that an application can write;
 - the need for atomicity of updates to preserve data integrity.

These issues can be properly dealt with by a persistent data API.

Recommendation: it is recommended that this API supports read/write properties. Overall, due to its simplicity, read/write settings storage is

considered a valued capability for the initial BONDI release. The complications mentioned above may be more robustly addressed through a persistent data API, yet such a standardised API is not available.

16.4 KEY ISSUE 3

Level of support for structured data: An API is envisaged that allows retrieval of values based on property keys. The question here is the degree of support for structured data, in terms of:

- The data types that may be represented as values. For example, the use of only strings or also primitive numeric types and structures etc.
- The level of structure within the key namespace, and whether or not there are operations available based on that structure (e.g. enumeration of properties within a given key subspace).

For value types, the argument in favour of supporting string values only is the simplicity of declaration of properties such as, storage and retrieval, and of the API itself. The arguments in favour of support for additional structure are:

- Efficiency: a value declared to be of a given primitive numeric type, for example, can be internalised as such which avoids the need for it to be parsed each time it is read;
- Code reuse: many applications will have some need to encode primitive type data, or structured data, within their settings, and providing support in the API avoids the need for application writers to reinvent a system to support them each time.

However, it is suggested that the additional complexity is quite considerable considering the relatively minor benefits.

Recommendation: it is recommended that only string value types are supported.

Regarding key structure, even with a totally unmanaged key namespace, it is quite common for users of a settings system to create implicit structure in their keys, each with the use of dot-separated key names, e.g. for settings in an e-mail program. For example, but not limited to:

- `server.pop.host = pop.myisp.com`
- `server.pop.username = blogs`
- `server.pop.password = letmein`
- `server.smtp.host = smtp.myisp.com`

The question here is if the structure is to be formalised by standardising the ‘.’ delimiter, and should there exist APIs that make use of this structure, such as a property enumeration API.

The argument in favour of having an enumeration API is that it is comparatively easy to support, and avoids the need for applications that potentially have many overwritable defaults to test separately for each property just to see if it has been defined.

The arguments against having an enumeration API are:

- Overall, due to its simplicity, read/write settings storage is considered a valued capability for the initial BONDI release. The complications mentioned above may be more robustly addressed through a persistent data API, yet such a standardized API is not available.
- Enumeration is impossible to implement efficiently on top of any underlying API that does not support enumeration. Therefore, in order to be confident that the BONDI APIs can be implemented as wrappers on top of proprietary APIs, enumeration should be avoided.
- Many of the use cases that motivate enumeration support can be handled in other ways by using a well defined sequence of property keys to ensure the keys are predictable, for example, but not limited to:

- `myproperty.count = 5`
- `myproperty.0 = red`
- `myproperty.1 = blue`
- `myproperty.2 = yellow`
- `myproperty.3 = orange`
- `myproperty.4 = black`

Recommendation: it is recommended that APIs that rely on property key structures are not included at this stage. However, the use of ‘.’ as a delimiter to create a hierarchical key space is reserved, to allow for the introduction of enumeration or other features in future.

16.5 KEY ISSUE 4

Privacy of properties: It is quite possible that a settings or configuration system will allow configuration of parameters that are meaningful to the framework but are not intended to be exposed to the application programmer. For example, a package manifest might include properties relating to application installation and security that are not intended to be visible to the application itself.

It is suggested that these properties are thought of as being conceptually separate metadata, even if ultimately they are represented in the same file alongside application settings. This means that it is necessary to ensure that they can be reliably identified as system data, separate from the application data, and it is not necessary to create a facility to indicate item-by-item which properties are intended to be readable by the application.

Recommendation: it is recommended that part of the key namespace is reserved for system properties to allow for implementations in which system and application property data may be encoded uniformly in a common container. It is suggested that the “bondi.” prefix is reserved for this purpose.

16.6 KEY ISSUE 5

Portability of representation: Although the textual representation of a property configuration (e.g. for use in a manifest or protocol exchange) is out of the scope of this document, it is important that the system is designed so that there is maximum flexibility in its representation. In this way the system does not place undue constraints on the future definition of textual representation. In particular, it is suggested that it is important to retain the ability to encode a property configuration naturally and portably in both a plain text and XML form, and to maximise the degree to which existing text processing code can be used to read property configurations.

Existing conventions relating to the use of a recognised key name delimiter and hierarchical key space are perfectly suitable provided there are certain restrictions, such as the requirement to use unique keys. Otherwise, there would be a natural representation of multi-valued data in XML but not as simple key/value pairs. For example, if an e-mail program understood multiple accounts, it might be natural to encode its settings in XML as:

```
<account>
  <name>Personal</name>
  <pop>
    <host>pop.myisp.com</host>
    <username>bloggs</username>
  </pop>
  <smtp>
    <host>smtp.myisp.com</host>
  </smtp>
</account>
<account>
  <name>Work</name>
  <pop>
    <host>mail.acme.com</host>
    <username>joe.bloggs</username>
  </pop>
  <smtp>
    <host>mail.acme.com</host>
  </smtp>
```

```
</account>
```

Alternatively, a representation as simple key/value pairs in plain text would be:

```
account.0.name = "Personal"
account.0.pop.host = "pop.myisp.com"
account.0.pop.username = "bloggs"
account.0.smtp.host = "smtp.myisp.com"
account.1.name = "Work"
account.1.pop.host = "mail.acme.com"
account.1.pop.username = "joe.bloggs"
account.1.smtp.host = "mail.acme.com"
```

Recommendation: it is recommended that the data model underlying the settings API is constrained so that it is easily and naturally representable in a range of formats. It is suggested that this is achieved by defining the key space as strings, implicitly hierarchical by adoption of the '.' delimiter, and each key is unique within any relevant scope.

16.7 REQUIREMENTS

REQ ID	HIGH LEVEL	MEDIUM LEVEL
IFC-630	A Web Application MUST be able to retrieve its configuration parameters.	
IFC-630.1		It MUST be possible to retrieve a single application property based on a provided key string.
IFC-630.2		The retrieved property MUST be returned as a string.
IFC-630.3		If a property is not defined, 'undefined' MUST be returned.
IFC-630.4		Any Unicode character is notionally valid in both a key string and a value string. Where particular characters need to be reserved for use in any particular encoding (e.g. '='), the definition of the encoding MUST also define an escaping mechanism to allow all characters to be represented.
IFC-630.5		The delimiter '.' in key strings MUST be reserved, in that the system reserves the right in the future to provide additional functionality that gives this character additional meaning.
IFC-630.6		Keys are expected to be uniquely defined in any given scope, and an implementation MAY treat a configuration as invalid if duplicate keys are present.

17 DEFINITION OF TERMS

TERM	DESCRIPTION
APPLICATION PROGRAMMING INTERFACE	Interface that is provided to the application in order to support the application's request towards the execution environment. In detail, an API consists of all classes, interfaces, methods, constructors, parameters, and exceptions that are required to provide the given functionality.
BONDI CAPABLE WEB PAGE	Application hosted on a web server that makes use of BONDI Interfaces.
BONDI EXTENSION INTERFACES	APIs compatible with BONDI architecture and not defined in the BONDI Interface requirements specification.
BONDI INTERFACE	API defined by BONDI
BONDI SECURITY PERMISSION	Capability that allows a Web Application to access all the methods and properties linked to that permission.
CONNECTION PROFILE	A set of parameters that defines the connection settings that the Browser or the Web Runtime Environment use in order to interact with remote servers.
EVENT PUBLISHING	Capability that allows applications to publish events so that subscribers to those events receive notifications about them
HYBRID APPLICATION	A Widget that has been through an Installation event but includes remote page
INSTALLED WIDGET	A Widget that has gone through an Installation event
LOCATION METHOD	Programmatic representation of the Location Resolution Procedure (e.g. Assisted GPS) used to obtain the location data
LOCATION PROVIDER	A programmatic representation of an individual hardware component (e.g. GPS component)
MEDIA GALLERY	Memory area in which media files such as images, video clips, sound clips are stored by default
SCRIPTABLE API	see BONDI Interface

TERM	DESCRIPTION
SCRIPTABLE BONDI API	see BONDI Interface
DEVICE	Used as an alternative term for a cellular telephone, terminal, mobile phone or handset.
WEB APPLICATION	An Application authored using web formats that makes use of Scriptable APIs, e.g. an installed Widget, web based Application or hybrid.
WEB BASED APPLICATION	Application hosted on a remote web server that makes use of Scriptable APIs.
WEB RUNTIME ENVIRONMENT	A software module that parses and executes browser formats (e.g. HTML and JavaScript), render web content to a display surface, and process user interaction (e.g., via keypad or touch screen) with that content. Installed widgets, web based applications, and hybrids will all typically use a WRE to render their content. Multiple Web Runtimes Environments may be installed on a Terminal
WIDGET	A Widget is understood to be an interactive single purpose Application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and Installation on a user's machine, mobile phone, or mobile Internet Terminal
WIDGET ENGINE	A user agent that checks the conformance of Widget Resources or its configuration documents, extracts file entries from within a Widget Resource's zip archive, renders various graphics formats, parses XML applications, manages a Document Object Model (DOM) and optionally supports HTTP, HTML, ECMAScript, CSS and other formats and specifications.

18 ABBREVIATIONS

ABBREVIATION	DESCRIPTION
AEE	Application Execution Environment
API	Application Programming Interface
APN	Access Point Name
CL	Communication Log
CSS	Cascade Style Sheets
DOM	Document Object Model
DPE	Device Profile Evolution
ECMA	European Computer Manufacturers Association
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Socket Layer
ID	Identifier
IMC	Internet Mail Consortium
J2ME	Java 2 Micro Edition
MIME	Multi-purpose Internet Mail Extension
MMS	Multimedia Messaging Service
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliance
OMTP	Open Mobile Terminal Platform
PIM	Personal Information Management
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol

ABBREVIATION	DESCRIPTION
SMS	Short Messaging Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
UICC	Universal Integrated Circuit Card
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WHATWG	Web Hypertext Application Technology Working Group
WLAN	Wireless Local Area Network
WRE	Web Runtime Environment
XML	eXtensible Markup Language

19 REFERENCED DOCUMENTS

No.	DOCUMENT	AUTHOR	DATE
1	RFC 2119 – “RFC Key Words”	IETF	
2	BONDI Architecture & Security Requirements http://bondi.omtp.org/1.0/security/BONDI_Architecture_and_Security_v1.0.pdf	BONDI	May 2009
3	“URI Schemes for the Mobile Applications Environment v1.0”	OMA	
4	“Wireless Application Protocol - Wireless Telephony Application Interface Specification”, WAP-268-WTAI-20010908-a	WAP Forum	2001-09-08
5	JSR-120 “Wireless Messaging API”	JCP	
6	JSR-75 “PDA Optional Packages for the J2ME(TM) Platform”	JCP	
7	Device Profile Evolution Technical Specification -	OMA	
8	RFC 2445 - “Internet Calendaring and Scheduling Core Object Specification”	IETF	
9	“VCard - The Electronic Business Card Version 2.1”	IMTC	
10	JSR-179 “Wireless Messaging API”	JCP	
11	“Overview of 2G LCS Technologies and Standards” - http://www.3gpp.org/FTP/workshop/Archive/0101LCS/Docs/PDF/LCS-010019.pdf	3GPP	

----- END OF DOCUMENT -----