Client-Side Enabler API Requirements

Candidate Version 1.0 – 27 Apr 2010

Open Mobile Alliance

OMA-RD-CSEA-V1_0-2010427-C

Contents

Figures

Tables

1. Scope (Informative)

This document defines the requirements for Application Programming Interfaces (API) to be defined as part of the OMA Client-Side Enabler API (CSEA) work item.

As a core feature of the "programmable Web" environment enabling Web 2.0, the CSEA work item focuses on the definition of API's. As an application environment capability, API's make client and server functions programmable, and change the market impact of what were previous siloed or inaccessible to applications and functions, by integrating them into Web 2.0.

The CSEA work item further focuses on the definition of API's as provided to client-side applications executing in Web runtime environments, e.g. in Web browsers or widget engines. The Web application environment provides a consistent application programming model through the use of Javascript as the primary procedural language through which application logic is coded and the CSEA API's will be defined. The OMA Browsing enabler is thus closely related to the CSEA enabler, in that a Web runtime environment such as defined in the OMA Browsing enabler is a foundational requirement for client-side Web applications.

The API's to be defined by CSEA are specifically scoped to enable Web application access to the services provided by OMA enablers as made available on the host device, e.g. through OMA enabler clients or other device software. In its initial release, the scope of CSEA 1.0 is further scoped to define the API requirements for the following OMA enablers:

- OMA enablers focused on access to content, as a typical web application need: initially limited to DCD, Push
- OMA enablers supplementing the web application experience with key capabilities enabling application personalization and contextualization: initially limited to DPE

Subsequent releases may be defined to extend this set of OMA enablers.

2. References

2.1 Normative References

[DCD] "Enabler Release Definition for Dynamic Content Delivery". Open Mobile Alliance™. OMA-ERELD-DCD-V1_0. URL:http://www.openmobilealliance.org/

[DCDTS] "Dynamic Content Delivery Technical Specification - Semantics and Transactions". Open Mobile Alliance™. OMA-TS-DCD_Semantics-V1_0. URL:http://www.openmobilealliance.org/

[DPE] "Enabler Release Definition for Device Profile Evolution ". Open Mobile Alliance™. OMA-ERELD-DPE-V1_0. URL:http://www.openmobilealliance.org/

[PUSH] "Enabler Release Definition for Push". Open Mobile Alliance™. OMA-ERELD-Push-V2_2. URL:http://www.openmobilealliance.org/

[PUSHCAI] "Push Client - Application Interface Specification". Open Mobile Alliance™. OMA-TS-PushCAI-V1_1. URL:http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[WebIDL] "Web IDL", Worldwide Web Consortium (W3C), URL:http://www.w3.org/TR/WebIDL/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version x.y, Open Mobile Alliance™, OMA-ORG-Dictionary, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 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].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Web runtime environment	Software which supports the execution of Web applications, e.g. browsers or widget engines.
Web application	An application that is written in Web technologies, e.g. HTML, CSS, Javascript.
widget engine	Software which supports the execution of Web applications running outside a browser context, e.g. with the same functional capabilities as browsers but without the user interface functions provided by a browser, including window frames, menus, toolbars and scroll bars.

3.3 Abbreviations

DCD Dynamic Content Delivery

DPE Device Profile Evolution

OMA Open Mobile Alliance

WRE Web Runtime Environment

4. Introduction (Informative)

4.1 Version 1.0

Version 1.0 contains the requirements for API's related to the OMA enablers: DCD, Push, and DPE.

5. CSEA 1.0 release description (Informative)

The CSEA enabler will define a set of Javascript API's to be made available for Web applications running in browser or widget context web runtime environments. The following diagram illustrates the role of the CSEA API's in device software. The API's are provided by the host web runtime environment, which may offer common execution support functions to application clients such as Web browsers or widget engines. The Web applications "run" inside the brower, as interpreted data and Javascript code, calling the CSEA-defined API's as the method to access OMA enabler functions. The Web applications can discover which CSEA-defined API's are supported in the host device, and then interact with the OMA enabler functions through the CSEA API's. The OMA enabler functions may be provided by the host web runtime environment, or accessed by it through API's provided by other device software (e.g. an OMA enabler client): this choice is out of scope of the WI and implementation dependent (in case of API's provided by software implementing OMA enabler client, the implementation of CSEA API's may go directly to those API's).

The diagram below shows an example in which the web runtime environment adapts its Javascript API to the interface/API provided by an OMA enabler client.

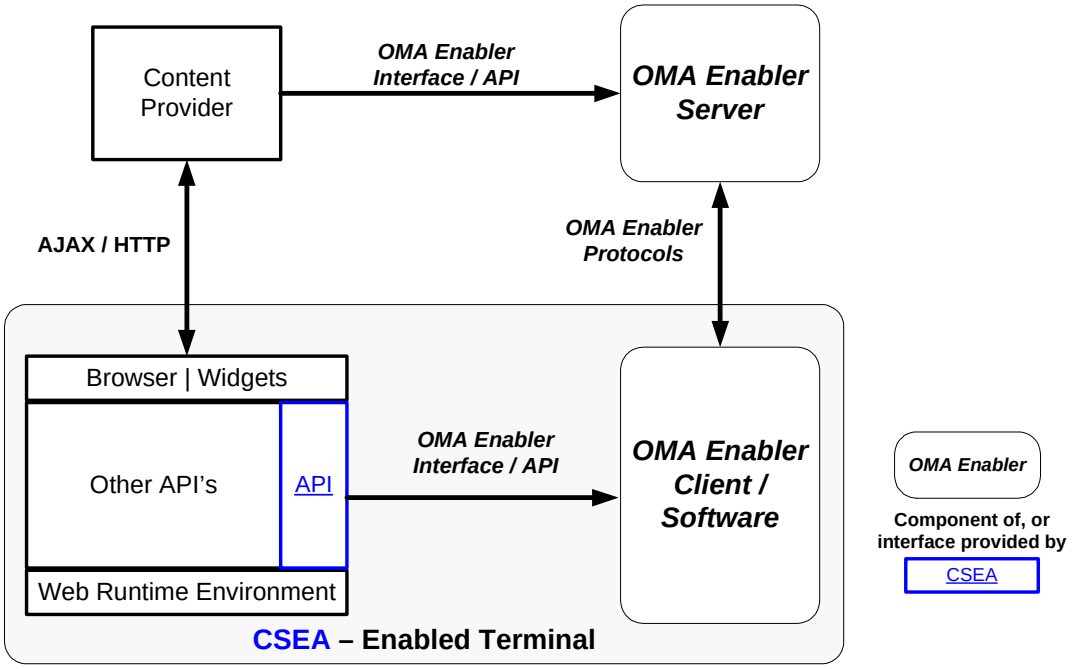

Figure 1 CSEA Descriptive Diagram

5.1 End-to-end Service Description

There is demonstrated interest in access to OMA enablers via client-side web application API's, i.e. Javascript-callable API's for browser and widget based web applications. This will allow the development of applications running on devices that can exploit functionality offered by OMA enablers, by accessing the related APIs on the device itself.

The functionalities of the CSEA enabler are not directly offered to the user, but are APIs offered to Web applications running on the device in either a browser or widget context.

6. Requirements (Normative)

6.1 High-Level Functional Requirements

Label	Description	Release
CSEA-HLF-001	CSEA SHALL support Javascript callable API's.	1.0
CSEA-HLF-002	CSEA API's SHALL be defined in Web IDL [WebIDL].	1.0
CSEA-HLF-003	CSEA API's support SHALL be discoverable by Web applications.	1.0
CSEA-HLF-004	CSEA API's SHALL be defined using consistent API design patterns, e.g. error handling, namespaces, and interface structure.	1.0
CSEA-HLF-005	CSEA API's SHALL support asynchronous operation.	1.0
CSEA-HLF-006	CSEA SHALL support API's to access OMA DCD enabler functions.	1.0
CSEA-HLF-007	CSEA SHALL support API's to access OMA Push enabler functions.	1.0
CSEA-HLF-008	CSEA SHALL support API's to access OMA DPE enabler functions.	1.0

Table 1: High-Level Functional Requirements

6.1.1 Security

Label	Description	Release
CSEA-SEC-001	CSEA API's SHALL be able to make use of the applicable security framework of the web runtime environment supported by the device.	1.0

Table 2: High-Level Functional Requirements – Security Items

6.2 DCD API Requirements

Label	Description	Release
CSEA-DCD-001	CSEA SHALL support the functions of the OMA DCD Client-Application Registration (DCD-CAR) interface.	1.0
CSEA-DCD-002	CSEA SHALL support the functions of the OMA DCD Client-Application Delivery (DCD-CADE) interface.	1.0

Table 3: DCD API Requirements

6.3 Push API Requirements

Label	Description	Release
CSEA-PUSH-001	CSEA SHALL support the functions of the OMA Push Client-Application Interface (CAI).	1.0

Table 4: Push API Requirements

6.4 DPE API Requirements

Label	Description	Release
CSEA-DPE-001	CSEA SHALL support Web application registration and deregistration with the DPE enabler.	1.0
CSEA-DPE-002	CSEA SHALL support delivery of property query requests to the Web application.	1.0
CSEA-DPE-003	CSEA SHALL support delivery of property status reports from the Web application.	1.0
CSEA-DPE-004	CSEA SHALL support delivery of property query requests from the Web application.	1.0
CSEA-DPE-005	CSEA SHALL support delivery of property status reports to the Web application.	1.0

Table 5: DPE API Requirements

6.5 Overall System Requirements

Label	Description	Release
CSEA-SYS-001	The CSEA Enabler SHALL leverage existing technologies and specifications defined by other standards bodies as far as possible to satisfy the requirements.	1.0

Table 6: Overall System Requirements

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version –or- No previous version within OMA

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-RD-CSEA-V1_0	14 Jan 2010	All	Initial baseline
	24 Feb 2010	6.5	Updated for agreed CR: OMA-MCE-2010-0004-CR_Standards_Reuse_in_CSEA_RD
Candidate Versions OMA-RD-CSEA-V1_0	27 Apr 2010	All	Status changed to Candidate by TP TP ref#: OMA-TP-2010-0183- INP_CSEA_V1_0_RRP_for_Candidate_Approval-

Appendix B. Use Cases (Informative)

B.1 Dynamic Content Delivery (DCD) API

B.1.1 Short Description

With the DCD API, Web applications that use ATOM/RSS content formats will be able to access that content through DCD, enabling simpler access to those services, and enhanced content delivery features. For example, developers can take advantage of DCD Push methods, which enable syndicated content delivery in ways not currently possible with conventional RSS/ATOM protocols, e.g. automatic network-initiated content delivery.

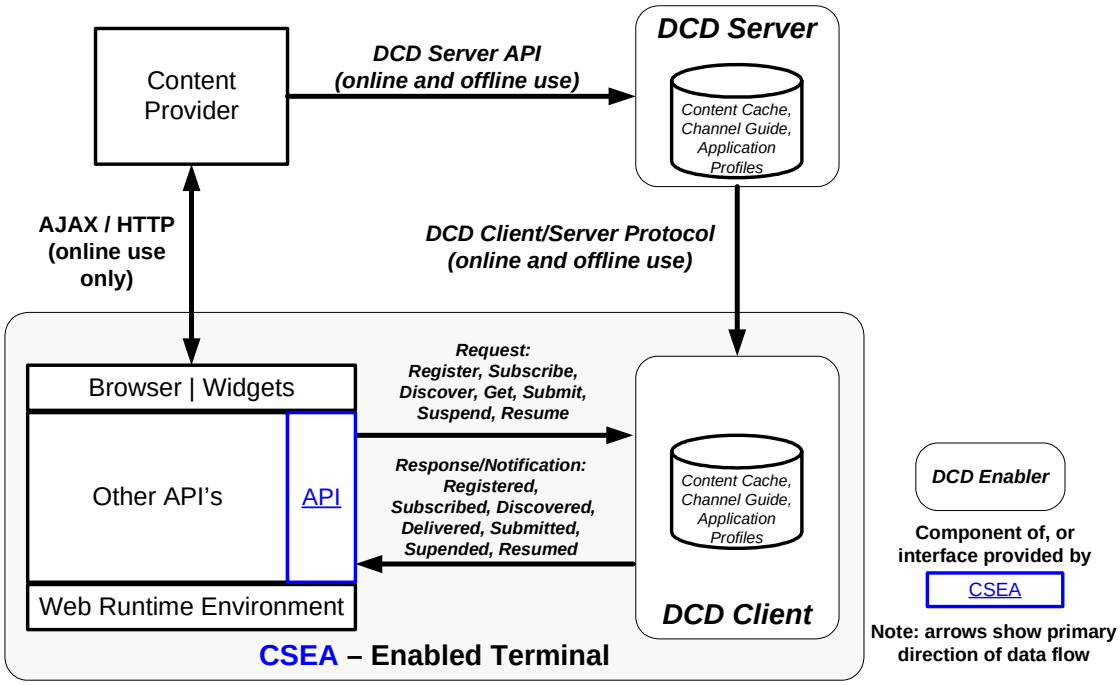

Figure 2 DCD Use Case for Web Applications

B.1.2 Market benefits

Offering DCD enabler services to Web applications will benefit application developers by:

- being able to create innovative mashup applications using content from many content providers, while avoiding the same-origin restrictions which otherwise would inhibit the development of mashup applications, by acting as a managed bridge to diverse content sources
- avoiding the need to implement the specific Ajax API's for diverse content sources
- avoiding the complications of deploying/running their own web application servers

B.2 Push API

B.2.1 Short Description

With the Push API, Web applications running in either a browser or widget context will be able to receive OMA Push events. Server-initiated content delivery via OMA Push is a widely supported and deployed service capability. OMA has defined the OMA Push enabler with specific support for certain applications included in standards (e.g. Web browsers, and specialized

clients for multimedia messaging, instant messaging, content download, Sync, Dynamic Content Delivery (DCD), etc), and with extensibility support for vendor-specific applications.

Many Web application examples can benefit from OMA Push support, e.g. for:

- Breaking news events
- Social network updates
- Scheduled updates for information services, e.g. web feeds
- Applications which may benefit from subscription to events over an extended period, and for resource efficiency are not expected to be always running, e.g. “watcher” type applications such as for stocks, auctions, shopping, security.
- Widget framework functions, e.g. advertising, content/application discovery.

B.2.2 Market benefits

Offering OMA Push enabler services to Web applications will benefit application developers by:

- enabling a content delivery method not currently supported for Web applications (server-initiated delivery), except through pre-standard, connection-oriented methods: Since the OMA Push API leverages signalling network capabilities, it can deliver connection-oriented or connectionless Push events using a standardized, widely supported set of protocols and client/server functions.
- supporting connectionless delivery of important application events, when the application is not running or does not have an active data connection: Connectionless delivery greatly improves the efficiency of device and network resource utilization.
- supporting connection establishment, e.g. delivery of “wakeup” events that cause an application to respond by establishing a network connection for delivery of further application data or processing
- supporting efficient delivery of connection-oriented events: Using OMA ush for delivery of application events over an established network connection, either during or outside of an active application session, avoids the need for applications to poll servers for events, while taking advantage of existing network connections for delivery of events directly to the targeted application. In addition, a single event delivery connection can serve multiple applications on the device, avoiding the device and network resource burden related to maintaining a separate connection for each server-sent event source.

B.3 Device Profile Evolution (DPE) API

B.3.1 Short Description

Via the DPE API, Web applications will be able to use the capabilities of the OMA DPE enabler, e.g. to disclose application capabilities/preferences/status to content/service providers in both online and offline use cases, enabling content and services to be matched to the application. DPE supports extensions to its core vocabulary, enabling access to application-specific properties. For example, the Delivery Context Ontology (DCO), a comprehensive and extensible characteristics data model for web applications developed by the W3C’s Ubiquitous Web Applications (UWA) working group, can be used through DPE to convey standardized properties and application-specific properties.

Similar to DCD, once an application has activated the DPE interface, the DPE enabler allows it to issue requests for properties, and the DPE enabler then delivers responses and unsolicited reports in response to prior requests. The properties can be those of the application or any other property that is supported by the DPE enabler. As exposed to client applications, this allows them to both publish their properties (by invoking the DPE API or by responding to a property query request via the API), and to query properties of the platform or of other applications, e.g. to obtain those which cannot be obtained through other platform API’s. The DPE API thus can enable the web application to issue a property request, and to respond to a request for one of its properties.

This diagram also illustrates the relationship between the DPE API and the other API’s which provide access to device properties, e.g. the Delivery Context Client Interfaces (DCCI) API (a draft specification of the W3C UWA), and the “Device

Status API" of the OMTP BONDI project, and the W3C's Device API and Policy (DAP) working group's related API (specification drafting just begun). While DCCI and the BONDI/DAP Device Status API are focused on application delivery of device properties directly to the Content Provider (e.g. as attributes in HTML forms, RESTful URI parameters, or other request data), the DPE API will enable property delivery via the DPE enabler, and also enable the Web application to obtain additional information about the device and its installed applications. These are complementary mechanisms, reflecting different use cases and deployment options.

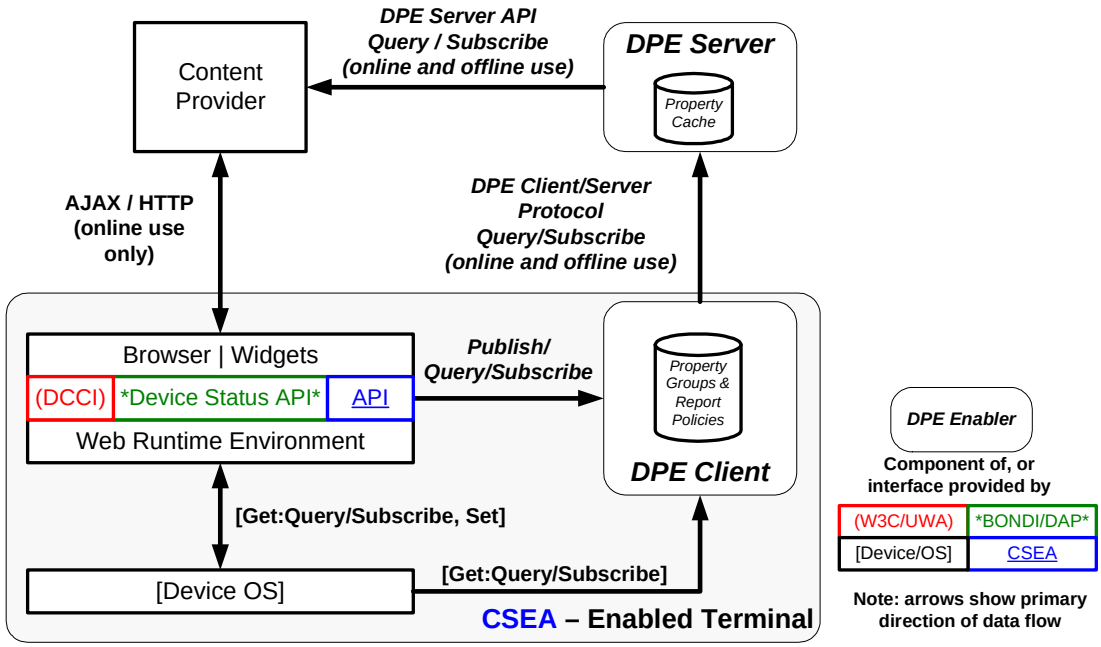

Figure 3 DPE Use Case for Web Applications

B.3.2 Market benefits

Offering DPE enabler services to Web applications will benefit application developers by:

- Providing the ability to publish their application's characteristics (e.g. application identity, capabilities, and preferences) through the DPE enabler, via which content providers can discover these characteristics, and thus tailor services offered to the application user in both online and offline use cases
- Enabling multiple content/service providers to be synschronized with the application's characteristics, without needing to synchronize with each and which may not be possible directly, due to same-origin restrictions
- Enabling content/service providers to be synchronized with the user's current device status, when the user switches devices