OMA Multimodal and Multi-device Enabler Architecture

Candidate Version 1.0 – 17 Jun 2008

Open Mobile Alliance

OMA-AD-MMMD-V1_0-20080617-C

Contents

Figures

1. Scope (Informative)

This document describes the architecture needed for the OMA multimodal and multi-device enabler.

Such a multimodal and multi-device enabler enables access to mobile services through different modalities (e.g. keypad, GUI, Voice, handwriting) or devices.

The architecture enables applications or services supporting multimodal or multi-device user interactions with minimal changes to the programming model for non multimodal and multi-device applications or services, e.g. browsing.

2. References

2.1 Normative References

[MMMD-RD] Multimodal and Multi-device RD, OMA-RD-Multimodal_Multi-device_Services-V1_1-20031112-A, November 2003.

[OSE] "OMA Service Environment"
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

2.2 Informative References

[ARCH-PRINC] "OMA Architecture Principles", OMA-ArchitecturePrinciples-V1_2, URL:http://www.openmobilealliance.org/

[ARCH-REVIEW] "OMA Architecture Review Process", OMA-ORG-ARCHReviewProcess-V1_4, URL:http://www.openmobilealliance.org/

[ETSI-DSR] *Aurora Distributed Speech Recognition, http://portal.etsi.org/stq/kta/DSR/dsr.asp*

[IETF-SIP] *Session Initiation Protocol, http://www.ietf.org/html.charters/sip-charter.html*

[IETF-SPEECH] *Speech Services Control (speechsc), http://www1.ietf.org/html.charters/speechsc-charter.html*

[OMA-DICT] "OMA Dictionary", OMA-ORG-Dictionary-V2_3,URL:http://www.openmobilealliance.org/

[W3C-MMI] *W3C Multimodal Interaction Activity, http://www.w3.org/2002/mmi/*

[W3C-MMI-REQ] *Multimodal requirements, W3C Multimodal Interaction Activity, http://www.w3c.org/TR/mmi-reqs/*

[W3C-DOM] *Document Object Model (DOM) Level 2 Events Specification http://www.w3.org/tr/2000/REC-DOM-Level-2-Events-20001113*

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.

This is an informative document, which is not intended to provide testable requirements to implementations.

3.2 Definitions

Backend	Application logic and supporting execution infrastructure that drives the presentation layer
Enabler	See [OMA-DICT].
Interaction and Multimodal Synchronization Manager	The logical component that coordinates data and synchronizes execution flow to and from the user agents.
Interface	See [OMA-DICT].
MMMD activation	Act of making a user agent available to a multimodal interaction and synchronization manager. Activation occurs within a MMMD session
MMMD de-activation	Act of making a user agent unavailable to a multimodal interaction and synchronization manager. Activation occurs within a MMMD session
MMMD discovery	Act of finding a component that matches the needs (capability, authorization, …) to support a MMMD interaction
MMMD Enabler	The OMA Enabler for MMMD user interaction
MMMD enabler registry	Addressable component where the MMMD components can register information. Some of these components can be their addresses for reachability through the multimodal synchronization protocol, or capability or services registered for etc.
MMMD registration	Act of making a component known and available to support MMMD interactions
MMMD service initiation	Act of setting components into the initial state when establishing a MMMD enabler configuration.
MMMD session	The logical connection between two or more MMMD components that are linked via MSP
MMMD session initiation	Act of establishing a MMMD session between two components: endpoints.
Modality	The type of communication channel used for interaction. It also covers the way an idea is expressed or perceived, or the manner in which an action is performed [OMA-MMI-REQ]. It can designate any channel where input or output interaction can occur.
Multimodal and Multi-device Configuration Protocol	Protocol to allow the MMMD capable end point to publish its own and discovery other end points multimodal or multi-device capabilities.
Multimodal synchronization protocol	Protocol responsible for the exchanges between the interaction and synchronization manager and the user agents that it synchronizes. These user agents may be on one or multiple devices
Processing Engine	The logical component that transforms user physical I/O to a format useful for the user agent (e.g. voice to text, ink to text, display, keyboard, T9, TTS…).
User agent	Logical component that can render presentation data into physical effects that can be perceived and interacted with by the user; And / or that can capture physical actions from the user and interpret them as interactions that can be reflected into presentation updates and/or update a data model. It may provide one or multiple modality and support input only, output only or combined input and output.

3.3 Abbreviations

IM/SM Interaction and Multimodal Synchronization Manager

MMMD	Multimodal and multi-device
MMCP	Multimodal and Multi-Device Configuration Protocol
MSP	Multimodal Synchronization Protocol
OMA	Open Mobile Alliance
OSE	OMA Service Environment
PE	Processing Engine
UA	User Agent

4. Introduction (Informative)

The purpose of this document is to present the architecture needed for the OMA multimodal and multi-device [MMMD] enabler and satisfy the requirements described in [MMMD-RD].

It describes how different modalities or devices can be synchronized so that interaction in one or multiple modality or devices is appropriately reflected in the other registered modalities and devices as well as in the application logic or data model.

It describes the different logical components required to support multimodal and multi-device interaction and the different deployment configurations that can be supported.

This architecture document does not assume particular authoring model for multimodal or multi-device interactions. The architecture supports the use of relevant standards from the W3C, IETF and others and identifies where OMA developed specifications are needed to create the desired open and heterogeneous computing environment. The architecture is not limited to any set of modalities or associated languages.

Figure 1**Error! Reference source not found.** shows the high level relationship of the multimodal multi-device enabler, specifically its role between user and backend functions executing multimodal and multi-device enabled applications. The MMMD enabler does not depend on any other OMA enablers. MMMD enabled applications may use other OMA enablers or resources through their exposed interfaces just as other enablers may use MMMD though its exposed interfaces. This document examines the componentry inside the enabler and, where necessary, specifies the required interfaces.

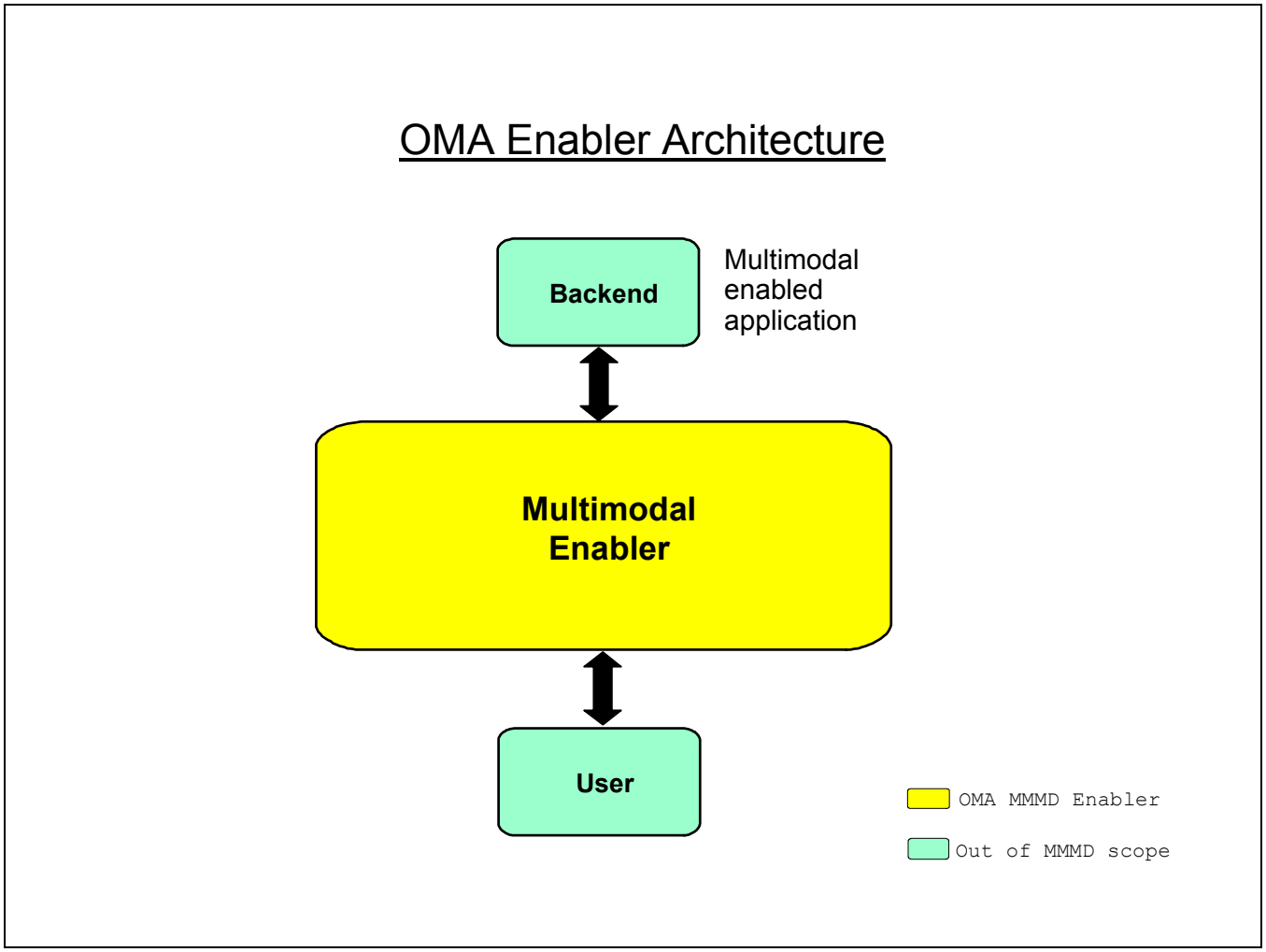

Figure 1: Enabler Architecture

4.1 Planned Phases

The MMMD architecture is expected to be fully defined in version 1.0. No additional phases are currently planned beyond version 1.0,

4.2 Security Considerations

The MMMD architecture does not introduce or need any additional security functionality. It supports the use of the security features already defined for browsing, e.g. transport layer security, authentication etc. There is no security functionality needed by the MMMD enabler beyond that defined herein.

5. Architectural Model

The MMMD Architecture is defined by its interfaces and its behaviour. The associated MMMD specifications will define interfaces; there are other interfaces which OMA need not define yet which are shown in the architecture. The MMMD architecture diagram, components and behaviour in different usage scenarios will be described in the following sections.

5.1 Dependencies

There are no identified dependencies for MMMD.

5.2 Architectural Diagram

Figure 2 shows the architectural model for the OMA multimodal and multi-device enabler.

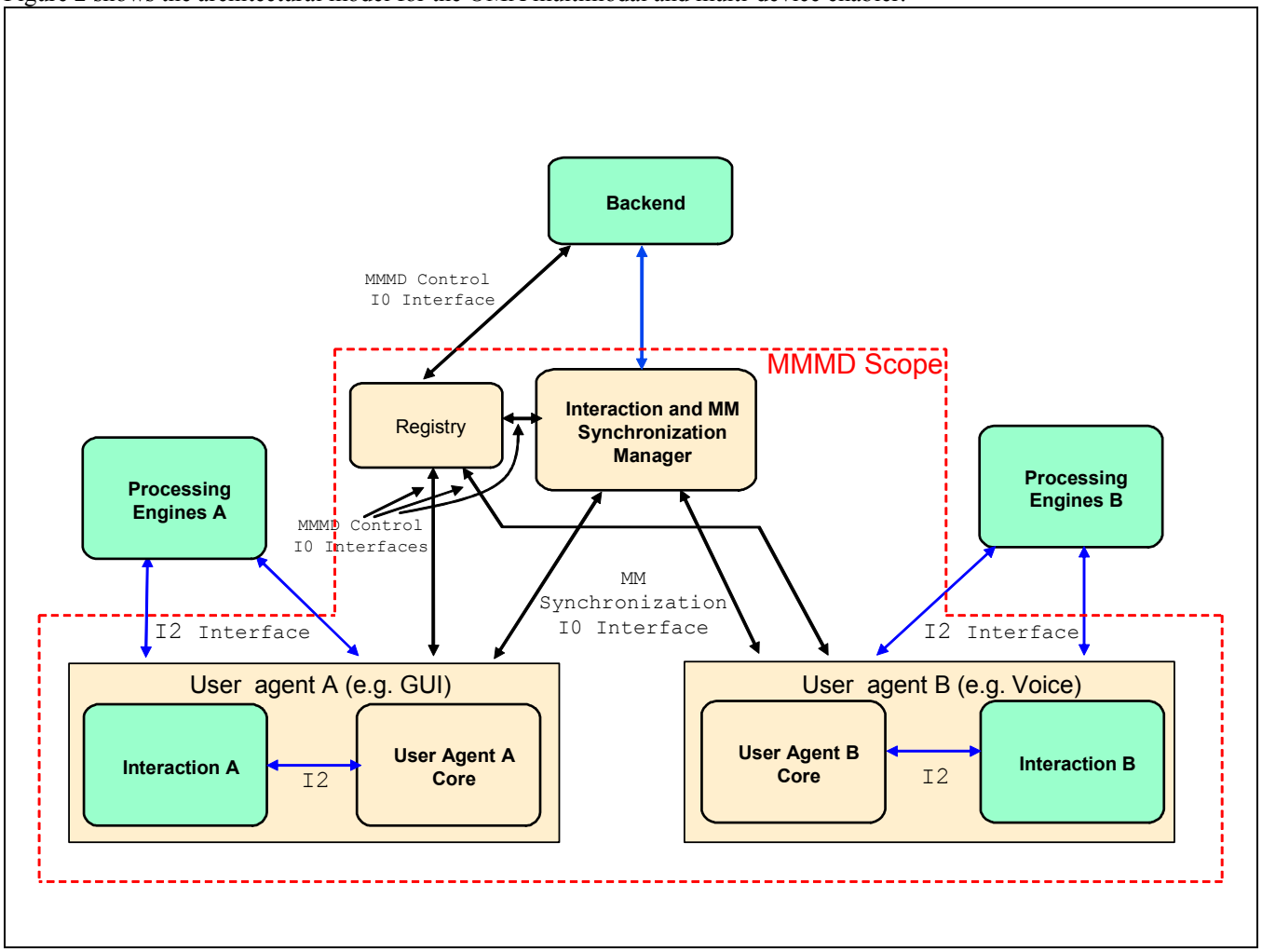

Figure 2: MMMD Architectural Model

The architecture shows the source of the interaction, e.g. keyboard entry, or presentation of graphics on a screen, separately from the main, or core, user agent functionality, the combination being considered a user agent for a modality, i.e. the GUI user agent. This functional decomposition permits a fuller understanding of the multiple sources of input and presentation and the factors MMMD has to address. In subsequent sections a more complete view with separate interaction capture, e.g. keyboard entry, from presentation is described with their respective processing engines.

The processing engines transform user input/output to a format useful for the user agent (e.g. Voice to text, ink to text, display, keyboard, T9, TTS…). Interaction capture takes the input (voice, keyboard, etc) from the user and translates the input into a form useful for later processing while interaction presentation converts the information from the user agent into a format that is understood by the end user.

The Interaction and Multimodal Synchronisation Manager function is to provide the means to synchronize the presentation and interaction captures from the different modalities being used.

The registry provides the means for applications and devices to indicate their multimodal, multi-device capabilities for discovery purposes etc.

The components of the architecture are described later in this document.

While shown in the logical architecture some components are deemed out of scope, i.e. the processing engines and the backend and associated interfaces. The rationale for this is as follows. The interfaces and function of the processing engines, e.g. voice recognition, is independent from MMMD, is likely to be vendor or platform specific in its capabilities and MMMD is only concerned with the generation of events and the generation of or presentation of data not the data itself. The interfaces to the PE depend on the platform and the vendors of the processing engine and user agent functions. The interface to the backend is not specific to MMMD but to the enabler that is MMMD enabled, e.g. the HTTP interface for browsing, where MMMD may require some additional information to be conveyed over the interface to the backend but the protocol remains intact and compatible.

Each of these logical components or portions of these components, i.e. processing engines, user agents or IM/SM, MAY be partitioned or combined on a single device (a piece of hardware that interacts with the user) or distributed across several devices or servers. Exemplary configurations are shown to illustrate some partitioning options. Devices may host MMMD user agents. Additional modalities can be added to a configuration by adding devices or user agents.

Figure 3 expands on the OMA Architecture from section 4 and begins to show how the logical components shown in figure 2 are positioned re the OMA enabler including the backend (i.e. application logic and supporting execution infrastructure that drives the presentation layer) and other OMA enablers to implement the MM execution model. The MMMD does not depend on other OMA enablers.

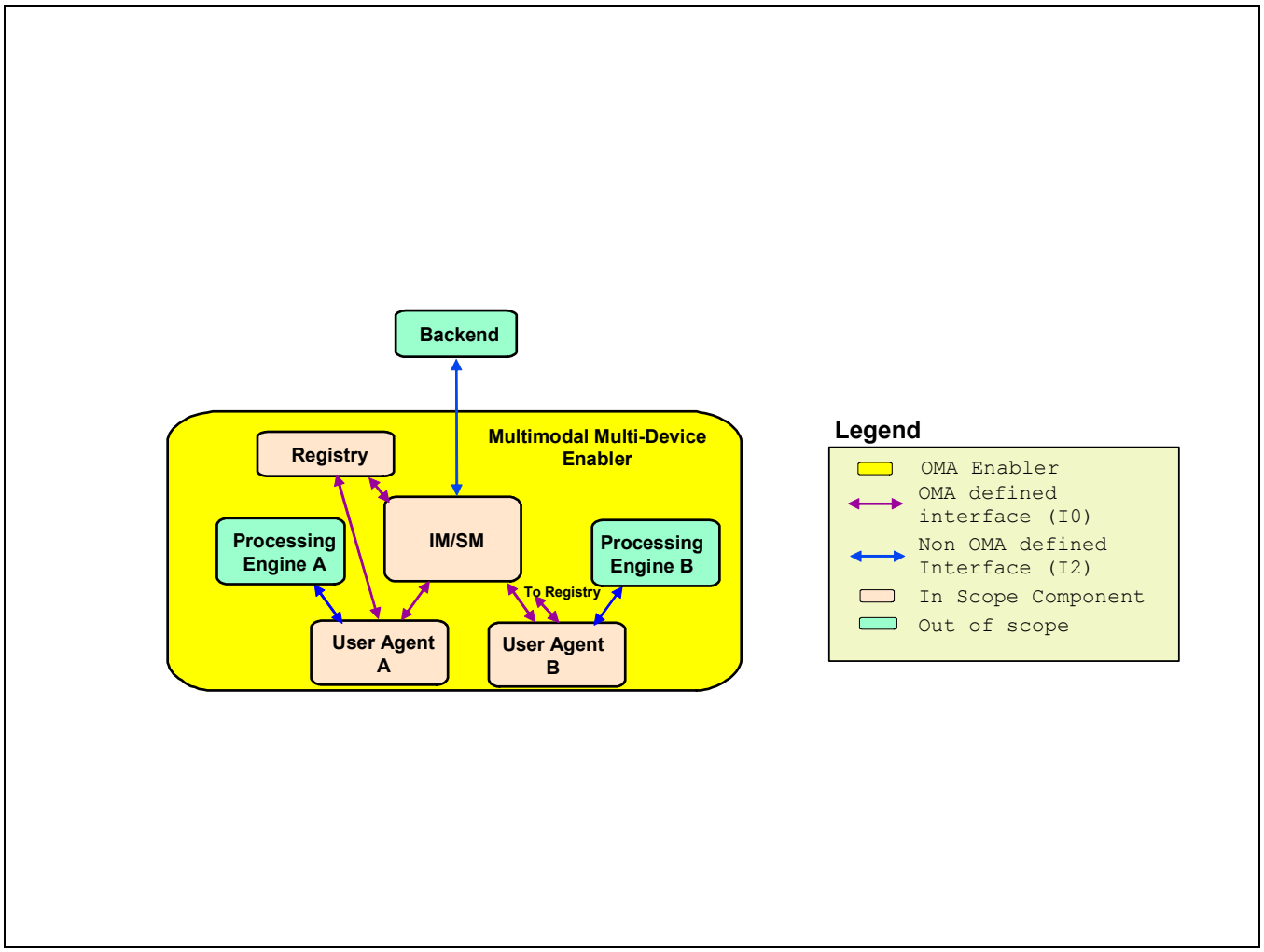

Figure 3: Logical architecture to support multimodal and multi-device interactions illustrated for 2 user agents (e.g. voice and GUI interaction

5.3 Functional Components and Interfaces

Figure 4 shows the logical architecture distilled from figure 2.

One or more OMA enablers, that may or may not have interfaces to them to expose their functionality to other enablers, make use of the IM/SM function via interface MMMD1 to perform the event synchronisation. Subsequently the IM/SM uses another OMA enabler, which may be part of the same enabler utilising MMMD1, via interface MMMD2 to perform the subsequent action. Once the necessary actions have been taken following the invocation via MMMD2, the response is provided via MMMD2 back to the IM/SM which manages the distribution of the response to the appropriate OMA enablers. The interface MMMD3 provides the means for MMMD deployed applications and user agents to register/deregister so their ability to support MMMD is discoverable and hence used; this is described more fully in subsequent sections. An interface between the user agent and the processing engine function is provided for reference purposes and shows how processing engines are interfaced to the UA but this is deemed out of scope since it would be defined either by the processing engine specification or that of the user agent using it.

Consider a more concrete example. In the case of voice or mouse/key based navigation in browsing at least one modality is used to present some information to the user. Whether voice based or mouse/key, an event is generated from the modalities that were used and an invocation of MMMD1 to the IM/SM made. The IM/SM then reconciles the events to form the overall event, e.g. selection of an <a> element with associated URL via whichever modality. The IM/SM then invokes MMMD2 to form the necessary next step, which in the case of the <a> element selection is a GET request to the appropriate URL.

Following the application server's actions the response, which may consist of at least one modality, is provided back through MMMD2 to the IM/SM where any action necessary, e.g. separating out the multiple modality presentations, is performed and the specific response to each presentation modality made through its MMMD1.

It is apparent in this example that MMMD2 may well be the normal HTTP interface for the browser. Moreover it is recognisable that MMMD1, even if HTTP, needs to convey information about the events that occur so the IM/SM can fully process these. Thus it is clear MMMD1 is something that needs to be defined, and will be the subject of some of the MMMD technical activity.

MMMD1, MMMD2 and MMMD3 are [OSE] I0 interfaces.

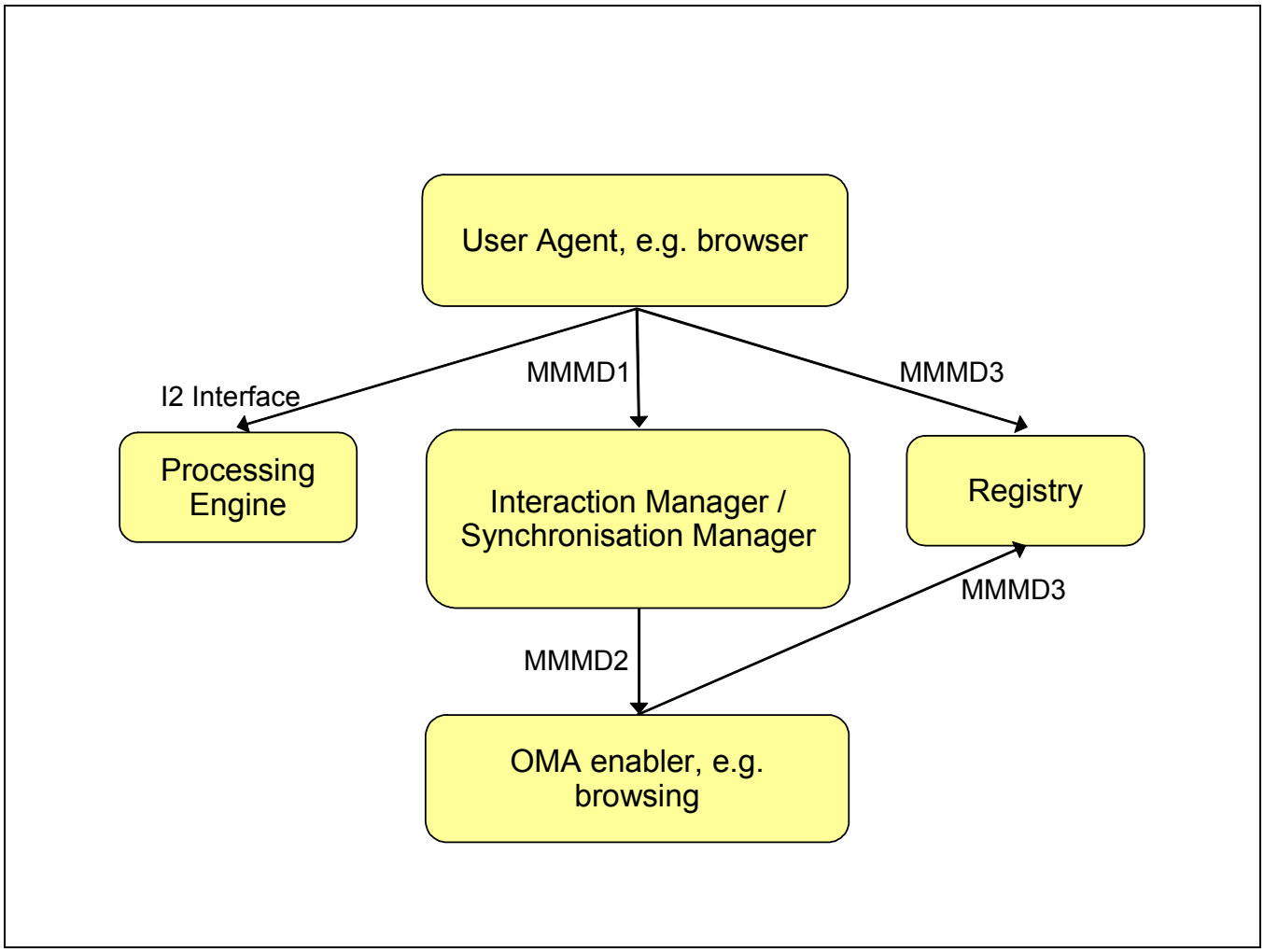

Figure 4: Logical architecture - OSE view

5.3.1 User Agent

In this section we take a closer look at the user agent and related logical components so we have a complete picture. With the understanding of this section in hand we are able to simplify the notation for user agent and thereby simplify diagrams. The W3C defines User Agent as "Any software that retrieves and renders Web content for users (see W3C Definition of User Agent, http://www.w3.org/TR/WAI-USERAGENT/glossary.html#u). This may include Web browsers, media players, plug-ins and other programs that help in retrieving and rendering Web content ".

In this document, we extend this definition and describe them as logical components that can render presentation data into physical effects that can be perceived and interacted with by the user; and / or that can capture physical actions from the user and interpret them as interactions that can be reflected into updates to the presentation and/or data model or trigger other events. It may provide one or multiple modalities and support input only, output only or combined input and output.

In the context of multimodality, user agents employ three important logical components:

1) Processing Engines Components - These engines may be input or output or both. In any case, they perform a transformation of the user physical I/O to a format useful for the user agent (e.g. voice to text, ink to text, T9, TTS …). Some modalities may not require such transformation and thus not need to use a processing engine.
2) Interaction Capture Components – This component captures interactions from the user in a form useful for later processing.
3) Interaction Presentation Components – This component presents the information from the user agent in a format that is understood by the end user.

Each of these components performs their roles in the context of the user agent according to the rules of the agent. Such rules can be built-in behaviours such as echoing typed text in a field or web content specified behaviours such as echoing text according to an application's style sheet. Not all permutations of input and output processing engines are always used and some components MAY be deployed in different places in various physical configurations. However, the relationships of all these logical components and the interfaces between them are shown in figure 5. Control and data MAY be flowing in all paths.

From an MMMD perspective most of these interfaces are not exposed and therefore are, as already described, considered [OSE] I2 interfaces. However the MMMD1 interface ensuring synchronisation of events between modalities and devices is key; thus is the principle user agent related [OSE] I0 interface.

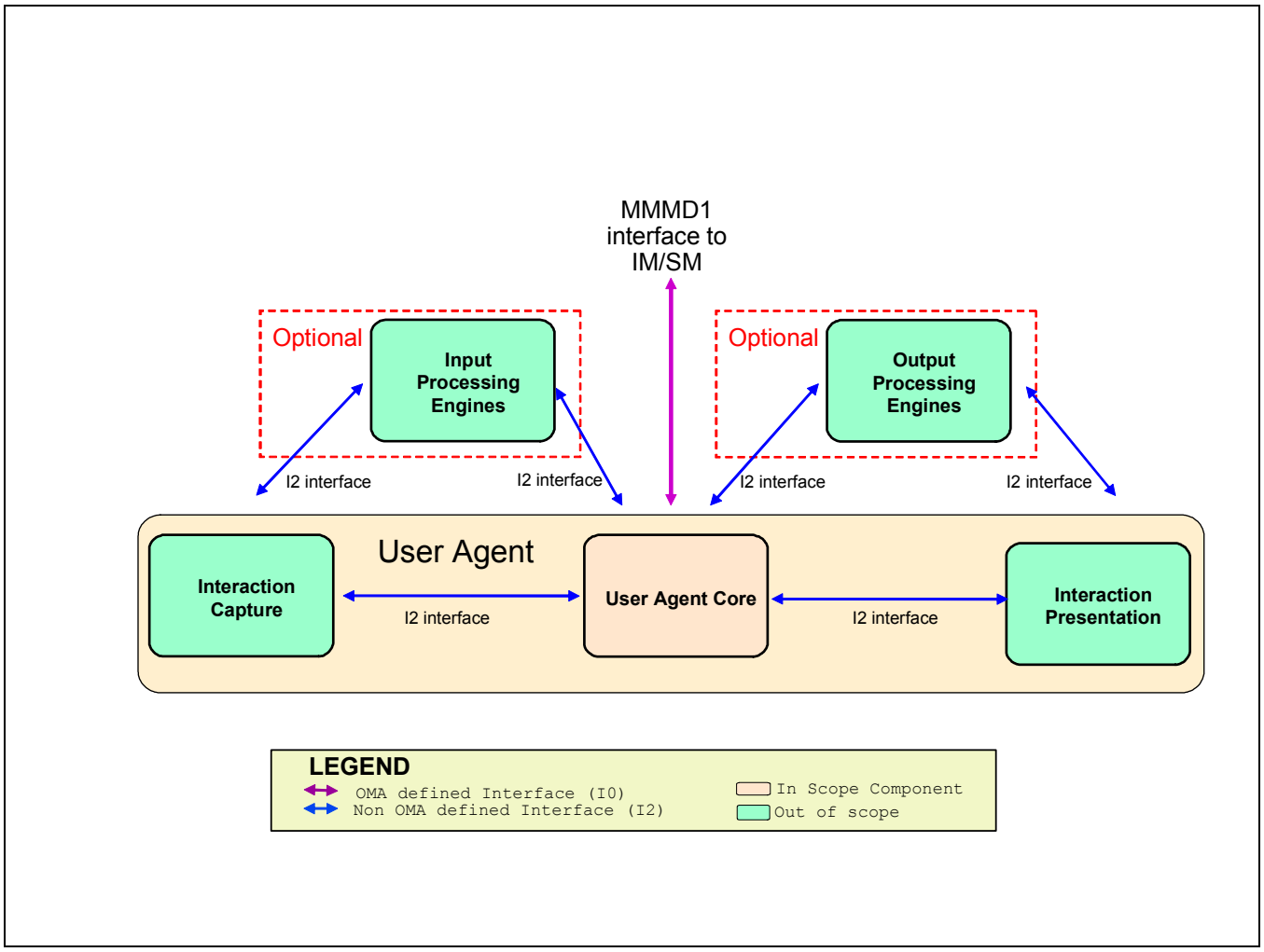

Figure 5: User Agent related logical components

The relationships between the UA and other components in figure 5 are flexible and can occur in a variety of configurations. To further help understand how the components work together exemplary configurations showing the optional aspect of the processing engines and the distribution capability of the capture and presentation with the processing engines are provided in section 5.3.5.

5.3.2 Interaction and Synchronization Manager

The Interaction and Synchronization Manager (IM/SM) is the logical component that coordinates data and synchronizes execution flow to and from the user agents. The IM/SM component implements the interaction and synchronization strategy of the application programming model. If required by the application model, the IM/SM may contain an interpreter for implementing the strategy. Compound configurations may codify IM/SM functions and multiple UAs in a single device using local proprietary interfaces. The important thing is that compound configurations implement the Multimodal Synchronization Protocol when interoperating with "external" components. The synchronization protocol supports both local and distributed components.

5.3.3 Multimodal Synchronization Protocol

The multimodal synchronization protocol is responsible for the exchanges between the IM/SM and the user agents that it synchronizes.

The interface I0 of user agents and the IM/SM supports the exchange of messages that:

- Provide notification of interaction events between the user agents (after capture, processing and interpretation by a user agent)
- Provide a mechanism to update the presentation of a user agent (to be delivered to the user possibly through processing engines and an interaction capture and presentation generation module).

The message may be signed and encrypted.

The Multimodal Synchronisation Protocol to be used in the MMMD enabler will be defined in a separate specification.

5.3.4 Multimodal and Multi-device Configuration Protocol

The MMSP synchronizes multimodal architectural components upon establishment of a particular configuration. Additional mechanisms are needed to support:

- Discovery of available/required modalities (distributed), including addresses of the user agent and its capabilities
- Registration, configuration selection and de-registration of the user agents
- Establishment of multimodal or multi-device sessions
- Activation and deactivation of user agents / modalities
- Life cycle management of the MMMD session

The Multimodal and Multi-device Configuration Protocol to be used in the MMMD enabler will be defined in a separate specification.

5.3.5 Exemplary Configurations

This section is informative

This section elaborates on the variety of ways that the logical components can be assembled to implement the RD requirements. This is illustrated through the exemplary configurations shown in figure 6.

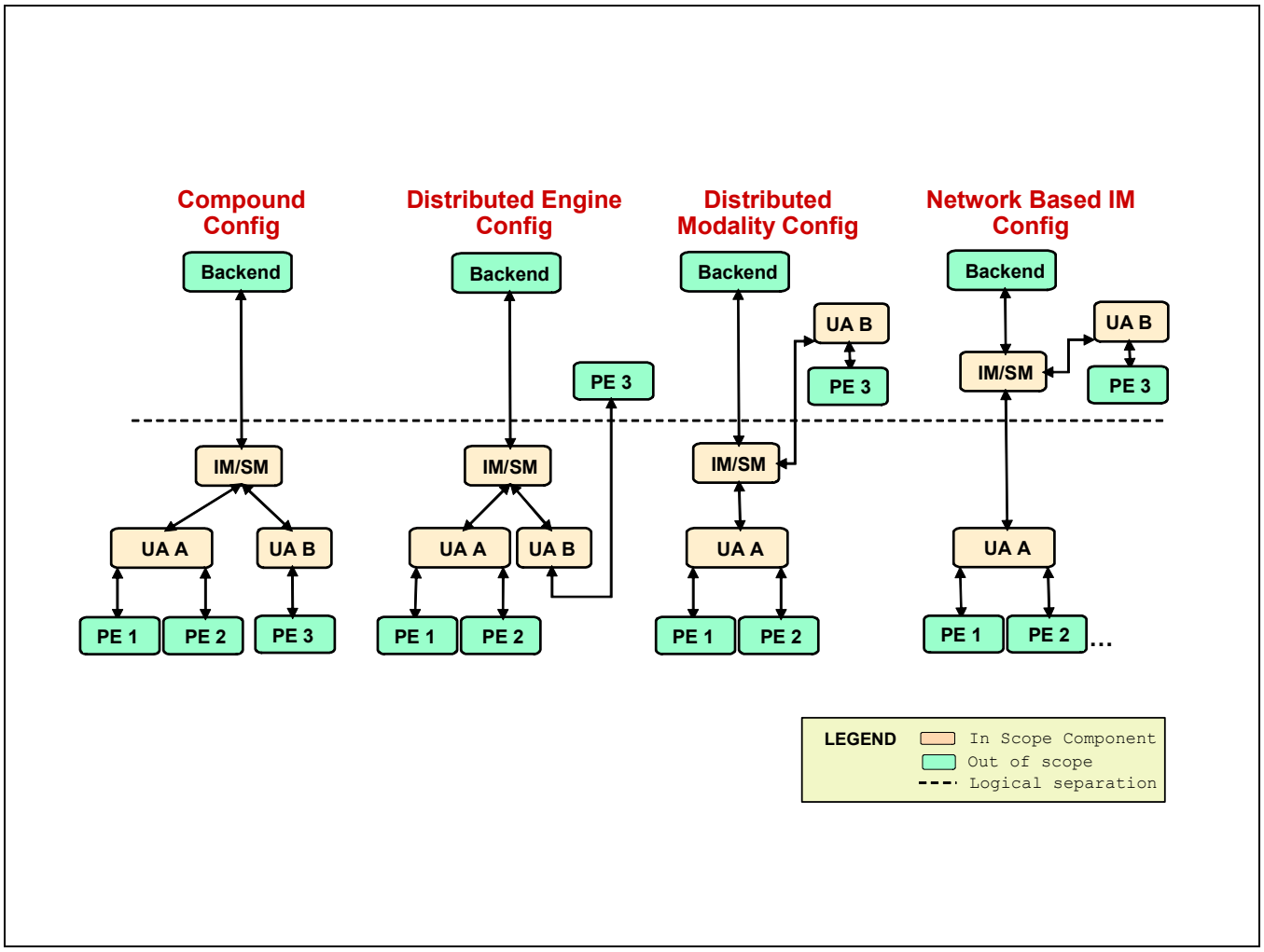

Figure 6: Exemplary Configurations of the MMMD logical components

While not an exhaustive list of permutations, the exemplary configurations show the various ways the logical components combine to deliver the required functionality. The dashed line represents a logical separation over which an IM/SM protocol is to be used; this may coincide with a mobile network, i.e. a classic device/server split. The MMMD enabler may be implemented in any of these or possibly others that are consistent with the logical architecture.

Compound Configuration

This is a configuration that fully implements all modalities within the device. Here, the device is shown with a User Agent A that codifies two processing engines into a single modality; this being illustrated by the traditional GUI with key/pointer input (PE 1) and text/graphical output (PE 2). User Agent B is shown illustrating another modality such as speech (with its own processing engine(s) PE 3) that is synchronized with User Agent A by the IM/SM. All together, this configuration illustrates multiple modalities implemented and synchronized in the device. The device may use a Browser, Java, Brew, or other device manufacturer provided environment as User Agent A. Since these GUI modalities may already have internal IM/SM capability, an implementation for a multimodal device that adds additional modalities such as User Agent B, may use a proprietary interface to integrate the additional user agent(s) with the GUI User Agent A. The backend is shown across the network but can also reside in the device.

Distributed Engine Configuration

This configuration is equivalent to the Compound Configuration except that processing engines are distributed across a network boundary into the network. This illustrates how a multimodal device may implement network based speech processing (PE 4) using a distributed speech capability such as ETSI Aurora or AMR-codec and IETF MRCP. As in the Compound Configuration, the GUI modalities may already have IM/SM capability so an implementation for a multimodal

device that adds additional modalities such as User Agent B, may use a proprietary interface to integrate the additional user agent(s) with the GUI User Agent A. The backend is shown across the network but can also reside in the device.

Distributed Modality Configuration

This configuration is similar to a Distributed Engine Configuration with the variation that a User Agent is distributed into the network. A key element of this example is that the IM/SM and User Agent C components are separated by a network boundary and thus illustrate the requirement for an externalized synchronization mechanism between the IM/SM and a distributed User Agent. While a synchronization mechanism exists in the compound configurations it may be an internal interface that is not generally externalized or standardized across ODM implementations for local modalities. However, a device supporting network based User Agents illustrates the need an externalized and standardized synchronization interface. This interface is of OMA interest. A device implementing this configuration must support a standardized external interface for distributed User Agents but may support local user agents using either an internal or external synchronization interface. That is:

1) A device may use an internal synchronization mechanism for local IM/SM and User Agent components (e.g. as in a Compound Configuration) and enable the addition of other modalities by implementing the external interface for distributed User Agents.
2) A device may separate IM/SM functionality from an existing GUI User Agent and implement the external interface to both local and distributed User Agents.

These variations are shown in figure 7.

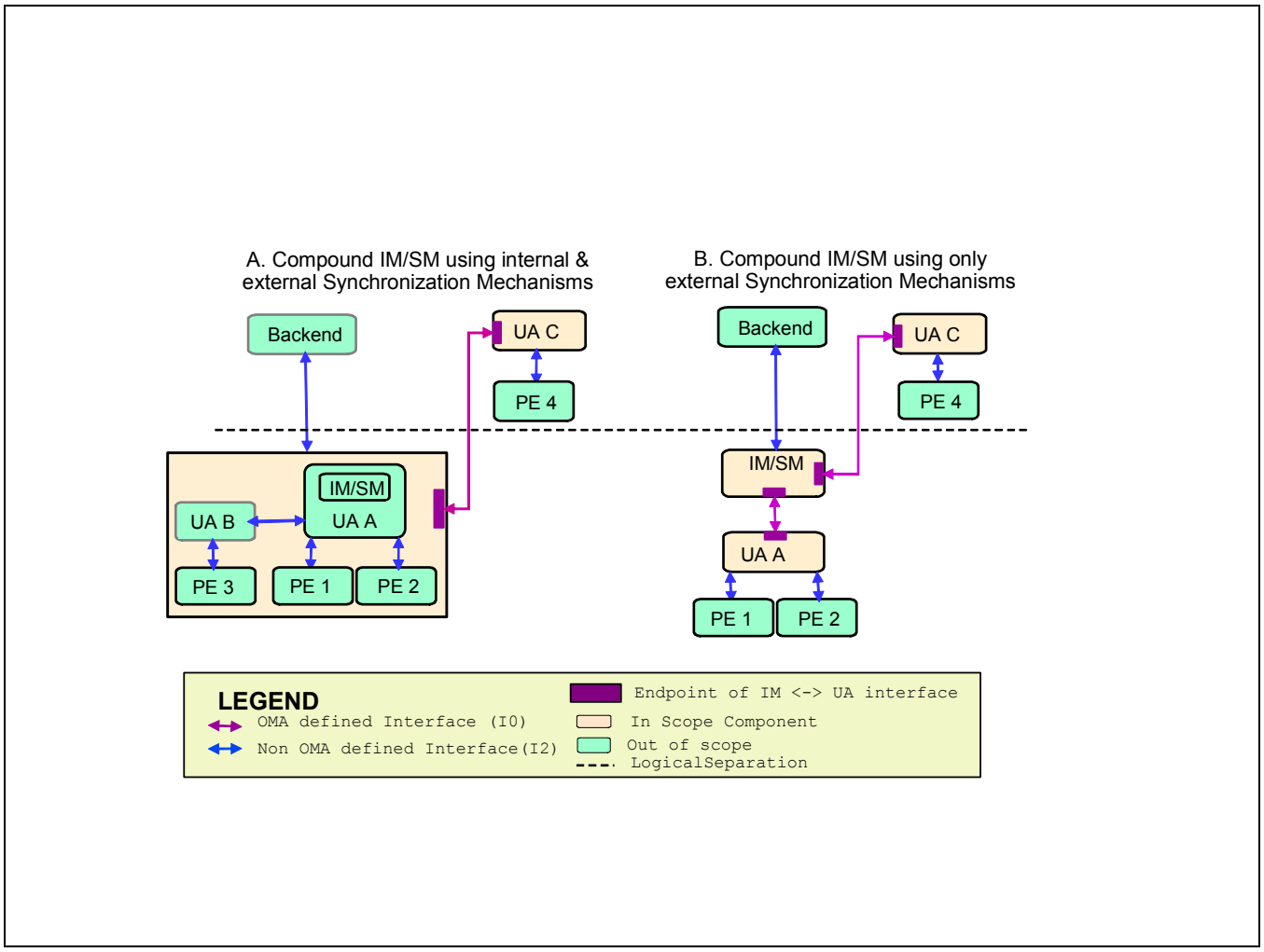

Figure 7: Variations of Distributed Modality Configuration

The variations shown in figure 7 illustrate how a single standardized synchronization protocol can allow different implementations of the device UI and device software architectures. In both cases, the enabler in the network is synchronized with the device UI without the network enabler being aware of the different device software architectures.

In configuration A:

1. User Agent A codifies 2 Processing Engines (e.g. keyboard & screen)
2. User Agent C has a single Processing Engine
3. Local Modalities are added to codified GUI using proprietary interfaces
4. User Agent C is synchronized with User Agent A using standardized external interface

In configuration B:

1. IM/SM component is separated from all User Agents
2. User Agent C is synchronized with User Agent A using standardized external interface

Network Based IM/SM

This configuration shows the IM/SM residing fully in the network. Here, the modalities are synchronized across network boundaries using a standardized external synchronization interface the same way as the Distributed Modality Configuration (i.e. a UA is distributed across the network from an IM/SM and synchronizes via a standard synchronization mechanism). Also similar to the Distributed Modality Configuration, the network based IM/SM may use the standard synchronization mechanism for "local" User Agents rather than an internal or private interface. These variations of this configuration mirror the variations of the Distributed Modality Configuration and are shown in figure 8.

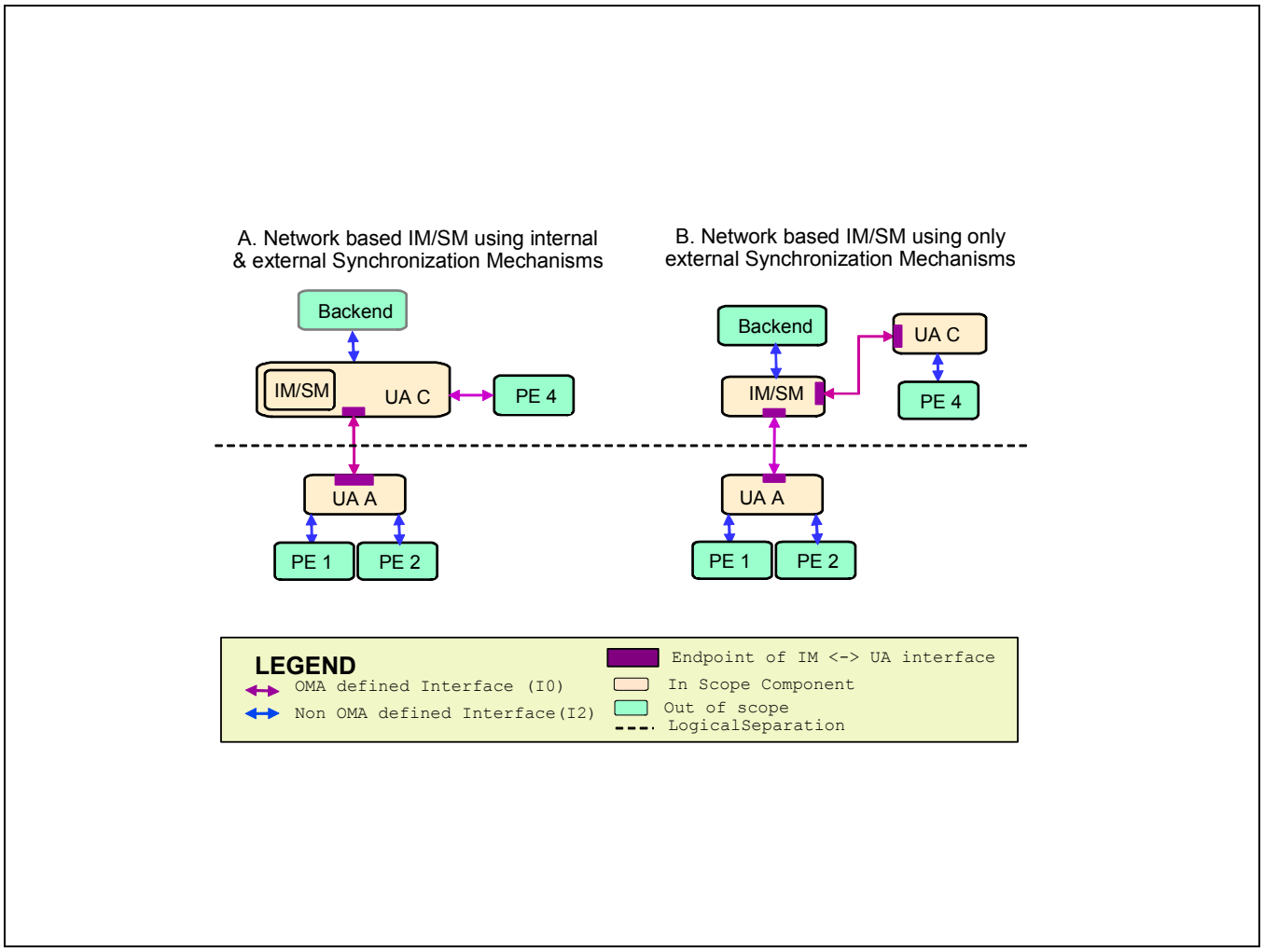

Figure 8: Variations of the Network Based IM/SM Configuration

These variations show the symmetry with the compound device configuration A in that the IM/SM function can be implemented within a network based User Agent and serve as the IM/SM logical component in either a master-slave or centralized model. They also illustrate how a single standardized synchronization protocol can allow different implementations of a Network Based IM/SM to work with any client implementing the same standard

User Agent exemplary configurations

To further help understand how the components work together, figure 9 shows three exemplary configurations showing the optional aspect of the processing engines and the distribution capability of the capture and presentation with the processing engines.

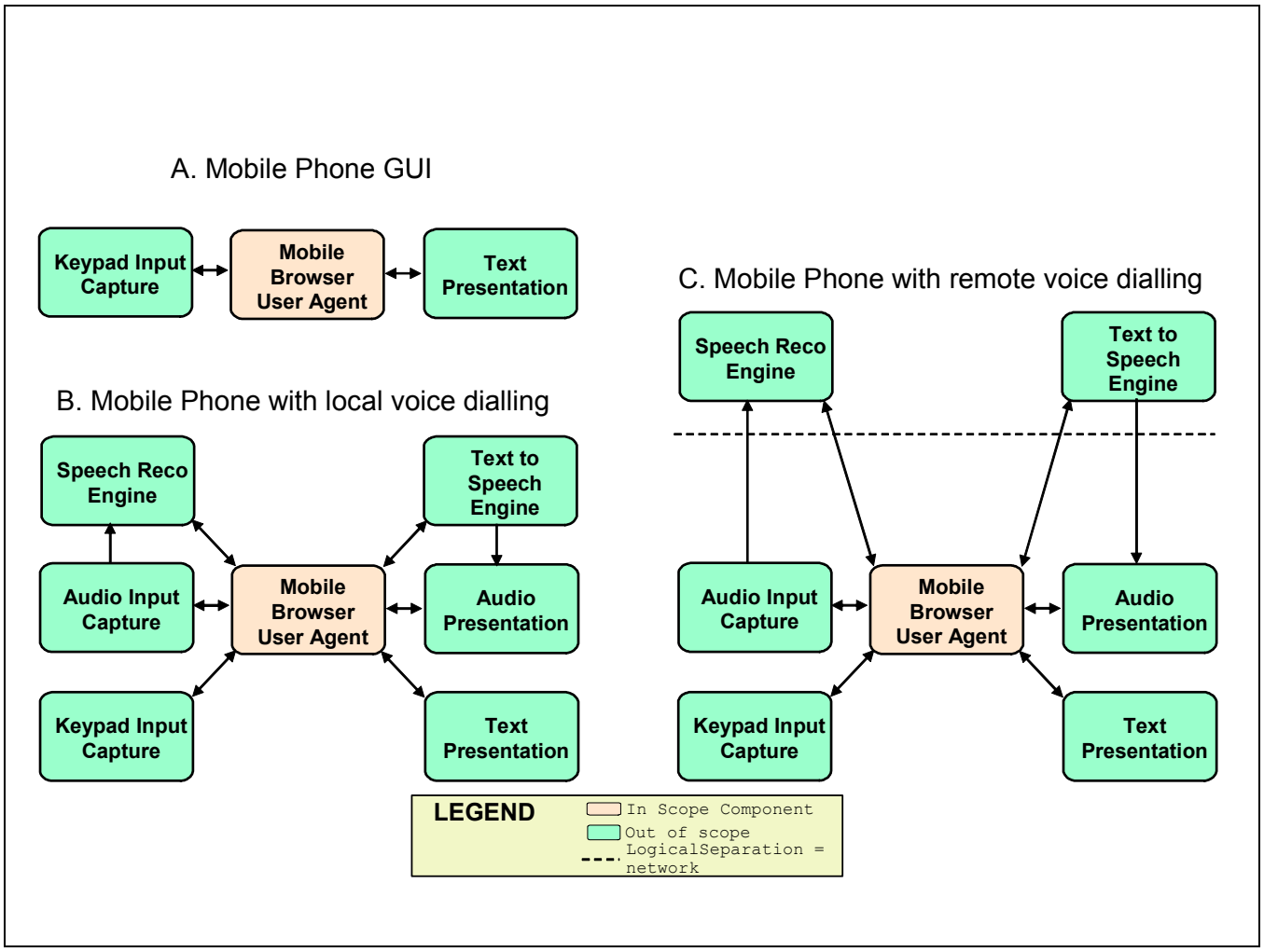

Figure 9: Exemplary User Agent Configurations

While the exemplary configurations show symmetry with engines and interaction components (e.g. figure 9.B), this does not always occur (e.g. a device may only have speech recognition engines and no TTS engines etc). However, when engines are present, there are typically two interfaces involved: data flow to/from the interaction component and control/results flow to/from the User Agent. To illustrate the interaction of these components, three scenarios are examined in sequence diagrams in figure 10 through 12. Figure 10 illustrates the roles of interaction capture, processing, IM/SM and user agent in a GUI only interaction.

In this notation, each modality is shown in the same column and which modality is inferred for a given action from the context. For example, an Output Presentation step of "Drawing a Red Button" is something done by the GUI modality. However, later in the diagram, an Output Presentation step of playing synthesized text is something done by the Voice modality. For diagram simplicity, both are shown in the generic Output Presentation column.

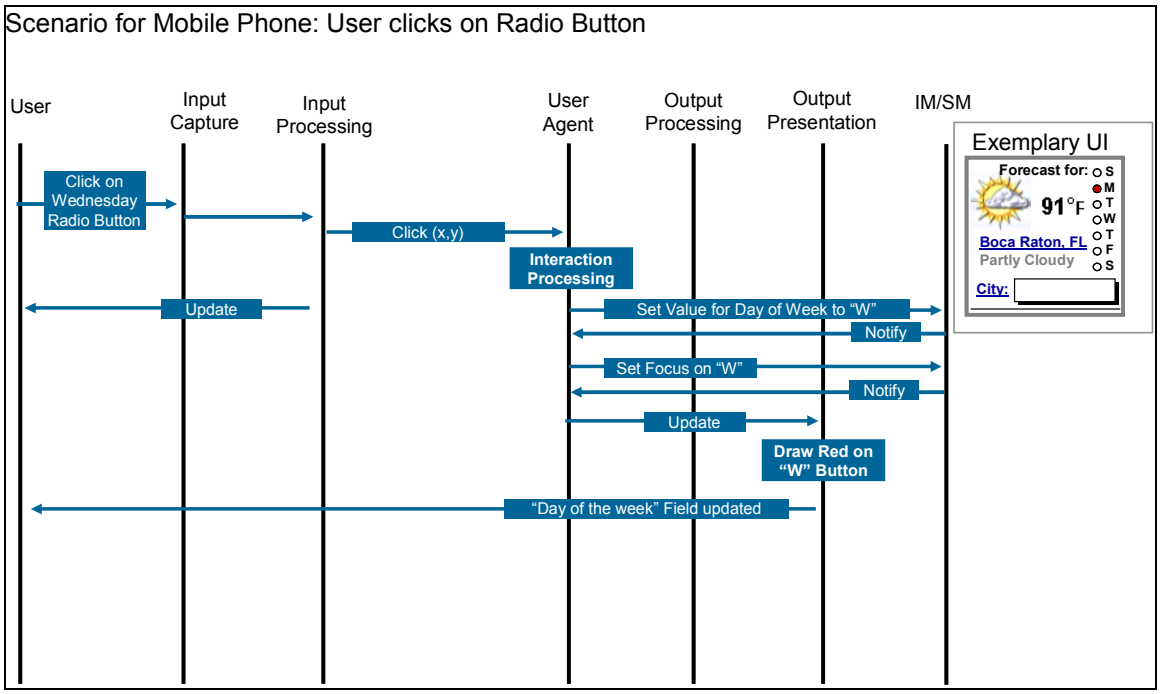

Figure 10: Scenario for Mobile Phone: User clicks on Radio Button

In this diagram, the typical events and processing that flow as the result of a typical action such as clicking on a Radio Button is illustrated. Input Capture handles the physical I/O and interfaces with the Input Processing subsystem which integrates the I/O into the GUI User Agent. After processing the click according to its internal rules and whatever application programming it performs, it takes the action to set the value of the Day Of The Week field to "W". In a multimodal system, that event is communicated to the other modalities via the IM/SM and so an event is sent there. The Notify step occurs for the User Agents that registered interest with the IM/SM and eventually, the GUI representation is updated by the Output Presentation layer to reflect the user input. This is an example flow of one possible implementation using the MMMD architectural components.

Figure 11 builds on this scenario by adding speech interaction to fill in a field. In this example, input focus is used to understand the context of the speech input. In this case, speech is used to fill in the city field in a weather application.

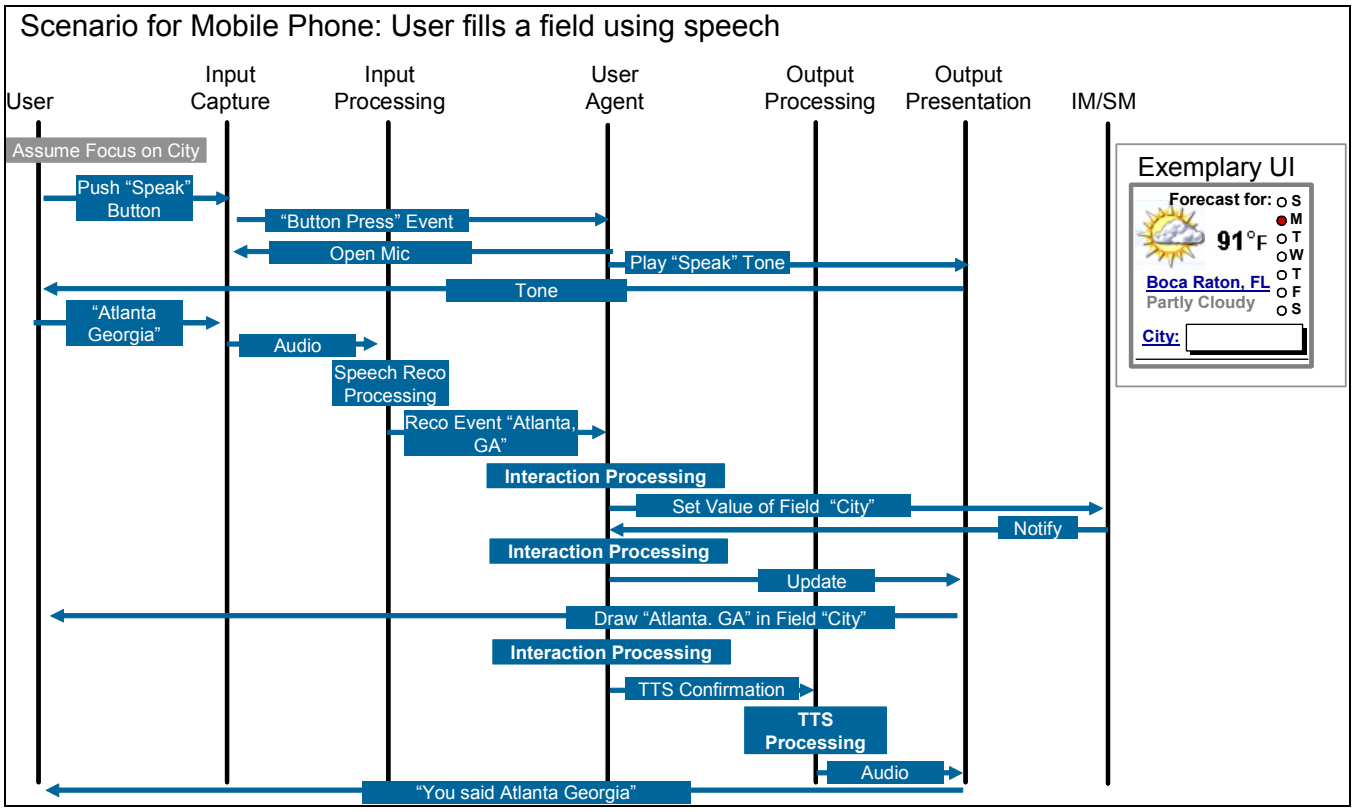

Figure 11: Scenario for Mobile Phone: User fills in field using speech

In this sequence, speech input capture is handled by the audio system of the device. The audio data is then examined in the processing layer (speech recognition in this example). Upon successful completion of the speech reco (recognition) processing, the recognized text is provided to the user agent handling the speech modality. Eventually, the speech modality user agent reflects the change in the value for the city field to the IM/SM so other interested user agents (modalities) can process the event and stay in sync. This is illustrated with the sequence of events by the second "Interaction Processing" step that, in this example, is a GUI user Agent that processes the fact that an input field was filled in via the speech modality and according to its processing rules plus that provided by the application, proceeds to update, in the example, the GUI. The update to the GUI is then done by the GUI Output Processing and Presentation layers as appropriate. The sequence finishes by illustrating the analogous (and potentially simultaneous) flow for speech output that starts with the user agent/application programmed response to recognized text and performs text to speech synthesis (processing) and audio output to the user (interaction).

Figure 12 takes these principles of the MMMD and illustrates an advanced form of multimodal interaction that incorporates simultaneous interaction in more than one modality that together provide the context for the application to interpret the user interaction. This is called deictic multimodal. In this scenario, the recognized text and GUI interaction combine to form a request for the weather on Friday. The city is not provided by the user and is inferred from the context of the current city (Boca Raton FL in this example).

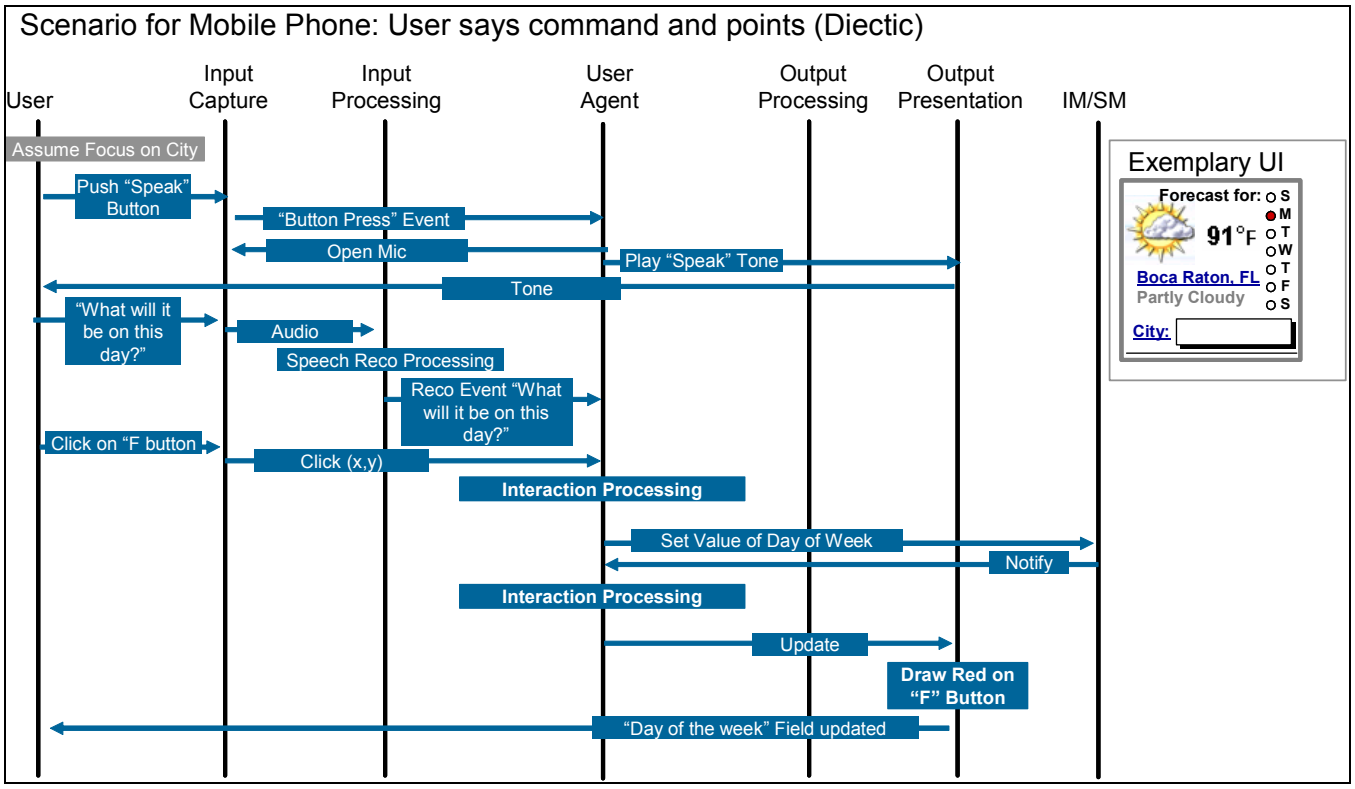

Figure 12: Scenario for Mobile Phone: User says command and points (Deictic)

This scenario uses the same principles of the previous scenarios where input interaction in a given modality is captured, processed and interpreted according to system and application provided programming. Similarly, output interaction is initiated, processed and presented in a given modality. Interpretation of user intent is ultimately determined by application programming with assistance from system layers with the IM/SM providing the necessary modality independent means to synchronize interaction between modalities. . There are other ways to process the initial event such as in the IM/SM.

Figure 13 shows two examples of the Multimodal Synchronization Protocol in use in compound configurations.

The MMMD architecture recognizes that coordinating interaction and coordinating Synchronization are important in a multimodal/multi-device system and that they are separate functions. Synchronization, as an example, is something that can happen at multiple levels and granularity depending on the configuration. Further clarity of these functions in terms of view and data synchronization/interaction and how they are each performed may be specified in future revisions of the AD.

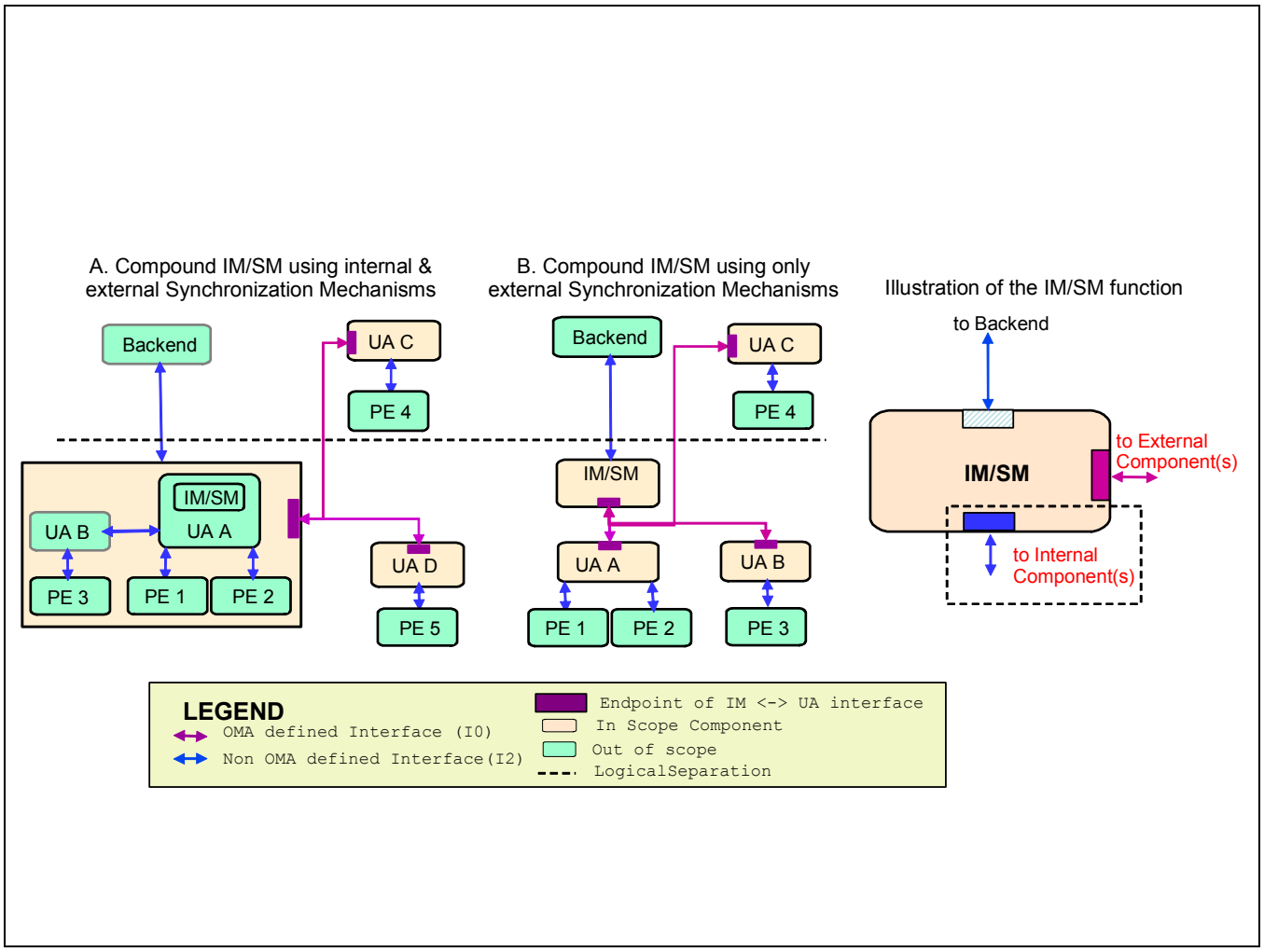

Figure 13: Examples of compound configurations with both internal and external components

Example A shows the architecture for configurations where the IM/SM is optimized for internal user agents A and B and uses the protocol for local user agent D and distributed user agent C. Example B shows the architecture for configurations where the IM/SM is a separate component using the protocol for local user agents A & B and distributed user agent C. Throughout the examples the term external implies the use of an standardised interface rather than an internal or proprietary interface.

5.4 Flows

The MMMD flows between architectural elements are described in general terms in this section. The detail of the flows is contained in the relevant technical specifications.

5.4.1 Example of MMMD configuration flows

We consider the following illustrative use case where user agents, upon access to a MMMD service will establish an appropriate MMMD enabler configuration as specified by the MMMD service[1]. The steps are illustrated in Figure 14.

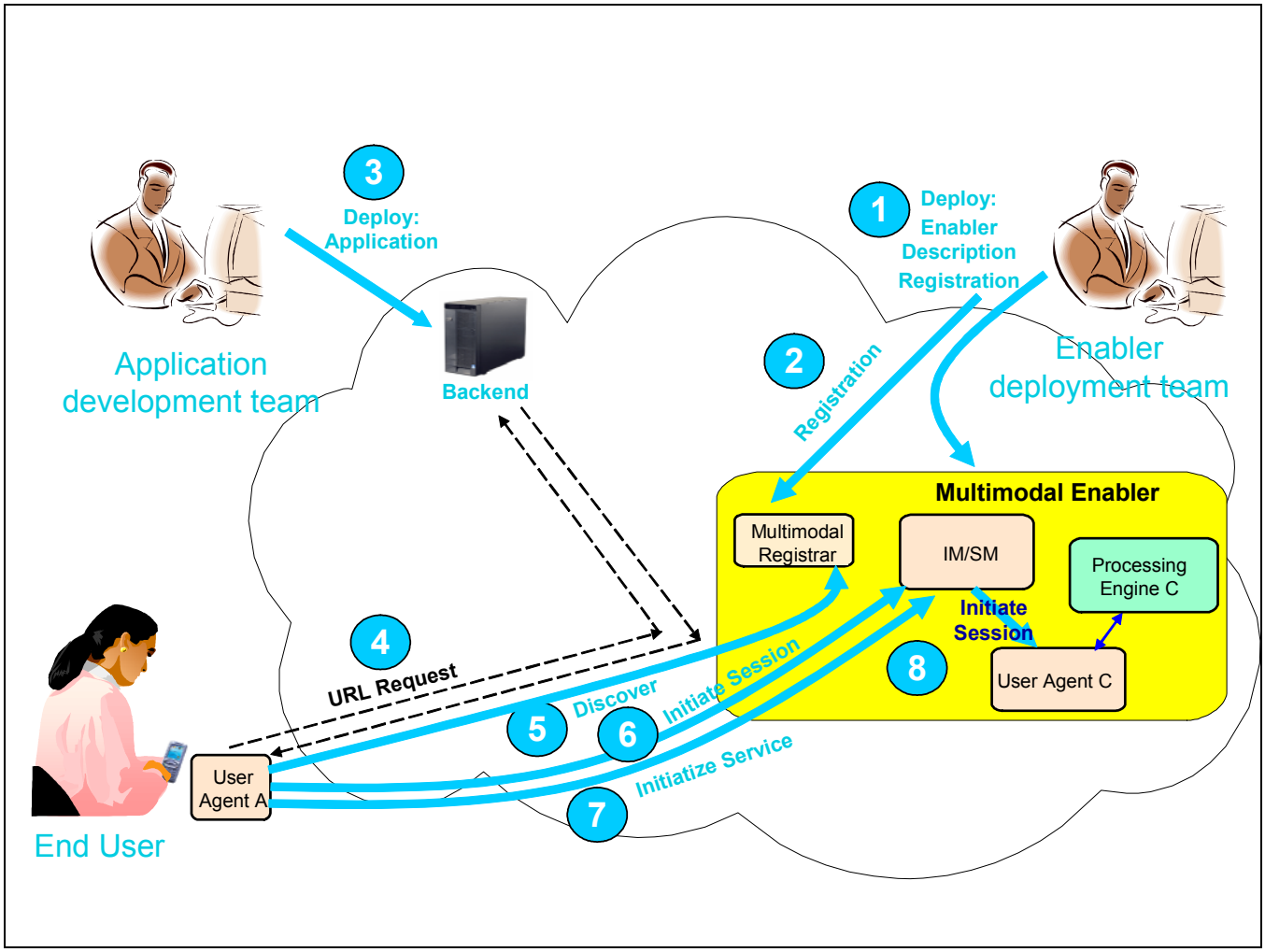

Figure 14 Flows that illustrate the configuration of a MMMD enabler implementation to support a MMMD application accessed via URI from a user agent.

1) Support for a MMMD service is deployed by a service provider deployment team.

2) This includes registering the addresses and capabilities of the different components that the service provider make available to support its MMMD services.

   Note: Registering follows the concepts of a SIP registrar.

[1] Note that other use cases involve having the configuration initiated by other components, possibly prior or independently of the MMMD service that is accessed. These other situations are also valid and expected to the supported by the MMMD enabler subject to ensuring the integrity of network security.

3) An application developer has written a MMMD application. This application is made available on a backend server (web application server), through a URI For the purpose of the present application, the application is assumed to describe the configuration and capabilities needed to its support.

   Note: For the sake of discussion, this use case assumes a web browsing model, hence the use of URI.

4) The user accesses the MMMD application by entering or clicking / selecting a URI on one of its user agents. The web application server executes the MMMD application.

5) The user agent that initiated the request determines the requirements in other components and configurations to support the MMMD service. The user agent discovers from the MMMD registry the available components. The service provider has made sure to provide what is needed for the service.

   Note: The user agent can determine the requirements in multiple ways. For example:

   - The application data could contain that information explicitly provided by the application developer (including possibly details on the required capabilities and/or configurations of the other components).
   - The user agent may be able to extract details on what other modalities, components and their capabilities to involve by parsing the received application data.
   - The user agent may be able to infer what other modalities, components and their capabilities to involve by parsing the received application data.

6) Upon discovering a candidate multimodal IM/SM, the user agent initiates a MMMD session with the assigned multimodal interaction and synchronization manager.

   Note: Logically, discovery of the candidate multimodal IM/SM is obtained from the MMMD registry. It is possible that this involves additional steps for example for load balancing of the different components.

7) After the MMMD session is initiated, the user agent initiates the multimodal service by providing the application data and its current state to the multimodal IM/SM.

8) The multimodal IM/SM uses this data to repeat itself steps 5) through 7) for all the other components that it needs to involve and synchronize.

Upon completion of these steps, the user agent is part of a MMMD enabler configuration, provisioned with the MMMD application. It is ready for presentation and interaction with the user.

The steps are illustrated in Figure 15.

Note that step 8 is the symmetric of steps 5 to 7 initiated by the multimodal IM/SM instead of the user agent.

5.4.2 Multimodal and multi-device execution flow

This section describes a basic flow for multimodal and multi-device interactions, to support the different use cases and requirements identified in the multimodal and multi-device RD. It supports the fundamental execution mode of multimodal and multi-device applications.

It is proposed that OMA multimodal and multi-device services satisfy this execution model.

The fundamental execution model of multimodal and multi-device applications is shown with red sequence numbers in Figure 15 and the following description:

1. A user interaction in one of the available modalities
2. The user agent implementing the modality optionally performs an immediate update to the user
3. Interaction event(s) resulting from step (1) are generated
4. The interaction events are passed to a IM/SM that handles a representation of the interaction event and determines the impact of the interaction based on the state of the application and synchronization scheme.
5. The IM/SM performs any required updates to its internal representation of state
6. Update events are sent to all the registered listeners (applications, modalities, …)
7. Any necessary presentation interaction resulting from step (1-6) is performed by the User Agent(s)

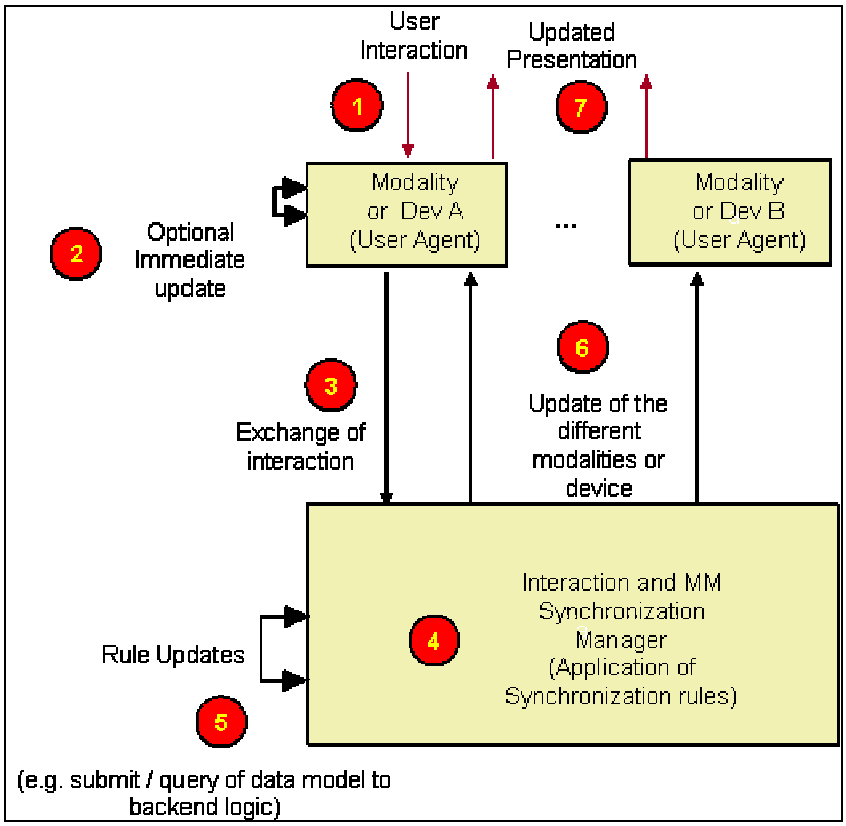

Figure 15: Fundamental execution model of multimodal or multi-device applications; independent of programming model or configuration.

This is summarized in Figure 15. In this figure, the User Agent and Multimodal Synchronization Manager logical components are diagramed to show how they implement the multimodal and multi-device execution model. Each user agent may represent a different modality (e.g. VoiceXML browser, XHTML browser, device GUI, or application) or different devices (e.g. smart phone and PDA or kiosk).

The user action may or may not result in an immediate update of the affected modality state prior to the synchronization (step 2). Immediate updates provide faster response in the modality that the user currently uses, but may lead to problems or confusing behaviors for example with composite inputs, concurrent or almost concurrent inputs in different modalities and conversational multimodal applications (i.e. where inputs are to be understood or disambiguated first). Handling this drives the need for additional logical components in the architecture for interaction capture/presentation in a modality (e.g. GUI I/O subsystem, audio subsystem…) and for processing user interaction (speech recognition engines, …).

5.4.3 Activation and Deactivation

During the MMMD session, the different modalities may be activated or deactivated by the multimodal interaction and synchronization manager. This may be based on:

- Preferences of the user made available to the multimodal interaction and synchronization manager (e.g. through a separate channel). For example this could include preferences on which modality to use a particular environment, with a particular device, at particular time, when particular presence status is set to particular values etc... . Activation / deactivation could also result from information passed from user agent as described below.
    - The IM/SM can stop sending multimodal synchronization messages and updates to deactivate user agents / modalities until they are reactivated based on preference or other mechanisms described in this section.
    - The IM/SM can also pass the deactivation information to the backend as part of the other mechanisms described in this section.

- The multimodal or multi-device application can provide presentation / update material related to activated modalities / user agents and hold any information related to deactivated modalities and user agents. The activation / deactivation is switched can be based on information available on the application (e.g. user preferences) or passed from IM/SM or user agents.
- Requests to activate or deactivate sent from a user agent (about itself or about another user agent / modality)
    - To the IM/SM that can react as for activation and deactivation by user preference available to the IM/SM.
    - To the application logic that can react as for activation and deactivation by user preference available to the application.
- The user agent may not be able to render its presentation material when deactivated and establishes de facto a new session with IM/SM when reactivated by the user.

Requests to activate and deactivate can be achieved by sending invitation to mute the communication associated to the user agent / modality to activate or deactivate. The mechanism is analogous to using SDP in SIP to mute a line by negotiating codec updates (see section 8.4 of RFC 3264): as for the session initiation a similar message allows activation and deactivation.

The requests to activate or deactivate may target a different user agent. Addresses resolvable in the MMMD registry are used to designate the target.

5.4.4 External events

External events are addressed by providing event listeners and handlers (e.g. see XML event model).

Event listeners and handlers may be provided at different levels:

- At the level of the user agents (and the presentation executed in the user agent)
- At the level of the IM/SM
- At the level of the application

Events are reflected as specified by the handlers.

The following mechanisms are available to pass events to their listeners:

- Registration of the listener to a resource when allowed by its I0 interfaces
- Registration to system events made available by the user agent, IM/SM or application execution environments.

Event listeners and handlers expression are dictated by the execution environment used to realize the enabler / presentation languages.

5.4.5 Preferences

Besides activation and deactivation, user preferences can influence what is made available in the different user agents and how it is rendered.

For now, preferences can be applied at different level:

- At the level of the user agents (how it renders a presentation)
- At the level of the IM/SM (how it transforms presentations and what it passes to different modalities / user agents)
- At the level of the application (what it associates to different modalities / user agents)

In the first release of the enabler the preferences are left proprietary to the vendor of UA, IM/SM and application execution environments. In future release these may be described in a standard XML format that may be exchanged / manipulate across the components.

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 V1.0 History

Document Identifier	Date	Sections	Description
Draft Version OMA-AD-MMMD-V1_0	15 Jun 2005	1,2, 3, 4,5	Incorporates input to date and put into current template
	24-Aug-2005	5.3, 5.4	Moved sections and added 20050212 contribution
	29-Aug-2005		Address comments from OMA-MAE-2005-0218-MMMD_Architecture
	05-Apr-2006	Many	Address informal review comments.(see ADRR and doc OMA-MAE-2006-0082)
	17-Jun-2006	5.3.5	New sequence diagrams added
	12-Jun-2006	5.3.5	Add diagrams and text from OMA-MAE-2006-0164R03-Sequence-Diagrams-for-AD. Source file for diagrams in OMA-MAE-2006-0124R04-SequenceDiagrams. Figure numbers auto renumbered and text aligned
	26 Sept 2006	Many	Incorportate OMA-MAE-2006-0293R01-CR_MMMD_to_address_ADRR_comments.doc which addresses AD review comments Remove Appendix B
	11 Oct 2006	3.3, 5.3.1, 5.3.5, 5..4.3	Incorportate OMA-ARC-2006-0346-CR_MMMD_AD_R_A_cleanup. Editorial change to 5.4.3. Correction to Date in history for OMA-MMMD-V1_0-20060926-D
Candidate Version OMA-AD-MMMD-V1_0	17 Jun 2008	All	Status changed to Candidate by TP OMA-TP-2008-0199-INP_MMMD_V1_0_RRP_for_Candidate_Approval Editorials: 2008 template, history box and definitions sorted