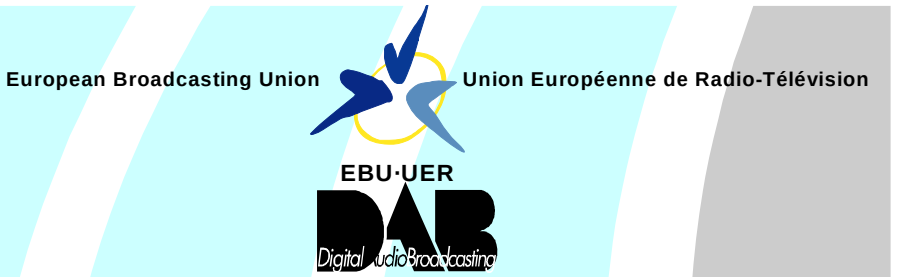
# ETSI TS 102 632 V1.1.1 (2008-11)

*Technical Specification*

## Digital Audio Broadcasting (DAB); Voice Applications




Reference
DTS/JTC-DAB-55

Keywords
audio, broadcast, DAB, digital, receiver, voice

***ETSI***

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

# Contents

# Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

# Foreword

This Technical Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE 1: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

The Eureka Project 147 was established in 1987, with funding from the European Commission, to develop a system for the broadcasting of audio and data to fixed, portable or mobile receivers. Their work resulted in the publication of European Standard, EN 300 401 [13], for DAB (see note 2) which now has worldwide acceptance. The members of the Eureka Project 147 are drawn from broadcasting organizations and telecommunication providers together with companies from the professional and consumer electronics industry.

NOTE 2: DAB is a registered trademark owned by one of the Eureka Project 147 partners.

# Introduction

The Broadcast WebSite (BWS) specification [1] defines a mechanism to format and transfer Web-like applications in the context of DAB. The BWS application is a relatively basic implementation of typical Web applications since it only offers a limited number of the usual capabilities.

Recent developments in mobile hardware have brought many new possibilities and further improvements in mobile devices can be expected for many years to come. These new capabilities could be used to provide a better and more complete user experience. In the case of the current BWS application, it is not well adapted to some mobility cases where the user is doing other things while using the system such as driving a car or crossing the street. Since BWS is mainly driven by a visual interface, safety and convenience in mobile environments could become important issues.

A more adapted system could thus be created to fill the lack of features of the old BWS system. The voice interface offered by new mobile devices could be used in the creation of new and enhanced services. Moreover, a new set of applications could be introduced to provide increased usability and user friendliness.

# 1 Scope

The present document describes Voice Applications specification. To begin with, the VoiceXML profile is explained with details about the modularization of VoiceXML 2.0, the dialog constructs, the user input, the system output, the control flow and scripting and the environment and resources. Then, the synchronization management mechanism is explained, followed by the specification of the extended interfaces. Finally, the transport and signalling of Voice Applications are detailed.

# 2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

For online referenced documents, information sufficient to identify and locate the source shall be provided. Preferably, the primary source of the referenced document should be cited, in order to ensure traceability. Furthermore, the reference should, as far as possible, remain valid for the expected life of the document. The reference shall include the method of access to the referenced document and the full network address, with the same punctuation and use of upper case and lower case letters.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

## 2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ETSI TS 101 498-1: "Digital Audio Broadcasting (DAB); Broadcast website; Part1: User application specification".

[2] ETSI TS 101 756: "Digital Audio Broadcasting (DAB); Registered Tables".

[3] W3C Recommendation: Voice Extensible Markup Language (VoiceXML) Version 2.0, 16 March 2004.

NOTE: Available at http://www.w3.org/TR/voicexml20/.

[4] XHTML+Voice Profile 1.2, 16 March 2004.

NOTE: Available at http://www.voicexml.org/specs/multimodal/x+v/12/spec.html.

[5] Standard ECMA-262, ECMAScript language specification, 3rd edition (December 1999).

NOTE: Available at http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[6] W3C Recommendation: Speech Recognition Grammar Specification Version 1.0, 16 March 2004.

NOTE: Available at http://www.w3.org/TR/speech-grammar/.

[7] W3C Recommendation: Speech Synthesis Markup Language (SSML) Version 1.0, 7 September 2004.

NOTE: Available at http://www.w3.org/TR/speech-synthesis/.

[8] W3C Candidate Recommendation: Semantic Interpretation for Speech Recognition (SISR) Version 1.0, 11 January 2006.

NOTE: Available at http://www.w3.org/TR/semantic-interpretation/.

[9] ISO/IEC 11172-3 (1993): "Information technology - Coding of moving picture and associated audio for digital storage media at up to 1,5 Mbit/s - Part 3: Audio".

[10] ISO/IEC 13818-3: "Information technology - Generic coding of moving picture and associated audio information - Part 3: Audio".

[11] ETSI TS 101 498-3 V2.1.1: "Digital Audio Broadcasting (DAB); Broadcast website; Part 3: TopNews basic profile specification".

[12] ETSI EN 301 234 V2.1.1: "Digital Audio Broadcasting (DAB); Multimedia Object Transfer (MOT) protocol".

[13] ETSI EN 300 401: "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers".

[14] IETF RFC 1952 (1996): "GZIP file format specification version 4.3".

[15] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) - Part 1".

[16] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) - Part 2".

[17] IETF RFC 2047: "Multipurpose Internet Mail Extensions (MIME) - Part 3".

[18] IETF RFC 2048: "Multipurpose Internet Mail Extensions (MIME) - Part 4".

[19] IETF RFC 2049: "Multipurpose Internet Mail Extensions (MIME) - Part 5".

[20] IETF RFC 4267: "The W3C Speech Interface Framework Media Types: application/voicexml+xml, application/ssml+xml, application/srgs, application/srgs+xml, application/ccxml+xml, and application/pls+xml", M. Froumentin, November 2005.

NOTE: Available at http://www.rfc-editor.org/rfc/rfc4267.txt.

## 2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area**.** For non-specific references, the latest version of the referenced document (including any amendments) applies.

Not applicable.

# 3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

| | |
|---|---|
| API | Application Programming Interface |
| ASR | Automatic Speech Recognition |
| BWS | Broadcast WebSite |
| DAB | Digital Audio Broadcasting |
| DMB | Digital Multimedia Broadcasting |
| DOM | Document Object Model |
| DTMF | Dual Tone Multi-Frequency |
| EPG | Electronic Program Guide |
| FIG | Fast Information Group |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| HTML | Hyper Text Markup Language |
| HTTP | Hyper Text Transfer Protocol |
| ISO/IEC | International Organization of Standardization/International Electrotechnical Commission |
| MIF | Mixed Initiative Forms |
| MIME | Multi-purpose Internet Mail Extensions |
| MOT | Multimedia Object Transfer |
| MPEG | Moving Picture Expert Group |
| PAD | Programme Associated Data |
| RFC | Request for Comments |
| SISR | Semantic Interpretation for Speech Recognition |
| SM | Sync Manager |
| SRGS | Speech Recognition Grammar Specification |
| SSML | Speech Synthesis Markup Language |
| TMC | Traffic Message Channel |
| TPEG | Transport Protocol Expert Group |
| TTS | Text To Speech |
| UA | User Application |
| UAT | User Application Type |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| USB | Universal Serial Bus |
| VA | Voice Applications |
| W3C | World Wide Web Consortium |
| WAP | Wireless Application Protocol |
| WWW | World Wide Web |
| XML | Extensible Markup Language |
| X-PAD | eXtended Programme Associated Data |

# 4 Introduction

In the context of DAB, Voice Applications are BWS applications enhanced with speech functionalities. They allow users to interact with new types of services with the help of efficient user interfaces such as spoken keywords, keyboards as well as text-to-speech rendering. These new "multi-modal" applications offer much more flexibility to the user in terms of navigation and information access. Voice Applications (referenced as VA throughout the present document) can thus be seen as an extension to BWS, considering that Web content can be presented with the visual and the audible interfaces concurrently. These new applications are possible thanks to the advancement of ASR and TTS technologies.

This goal is accomplished efficiently by profiling the VoiceXML standard [3] which was designed for interactive and voice responsive applications. Some minor extensions are also defined for DAB relevant elements covering synchronization management, a DAB interface, and a GPS interface. The synchronization management deals with concurrent execution along with BWS content. The DAB interface addresses the access mechanism required for the VA content to be integrated with the DAB resources. VA also defines the GPS elements to support location based services.

The delivery of VA is analogous to the delivery of BWS application. It can be downloaded to the end user device using the MOT protocol [12]. Interactive content can also be downloaded if the return channel is activated.

# 5 Voice Applications

## 5.1 Overall VA service framework

The VA system can be divided into sub-systems that interact together to provide voice enabled website applications to the end-user. As a first step, a VA application is transferred to the VA platform, residing on the end user device, through the broadcast or bi-directional channel. Then, the platform executes the application and, as a consequence, the execution is presented to the user through various output interfaces. The user can interact with the platform by using the available input mechanisms. An optional sub-system can be included to enable remote interaction between the platform and the broadcasting site. Remote interaction requires however the presence of a bidirectional communication channel for the provision of interactive services.

Figure 1 depicts the sub-systems of the overall VA system. The sub-systems presented with dotted lines are optional and the arrows show the direction of the data flow. As described in the following clauses, several actors are involved in a VA service framework and can be referred to as: the user, the broadcast site, the telecommunication site, the VA platform, the broadcast channel object acquisition module, the bi-directional channel acquisition module and the GPS module. The telecommunication site, the bi-directional channel acquisition module and the GPS module are optional actors in the VA service chain.

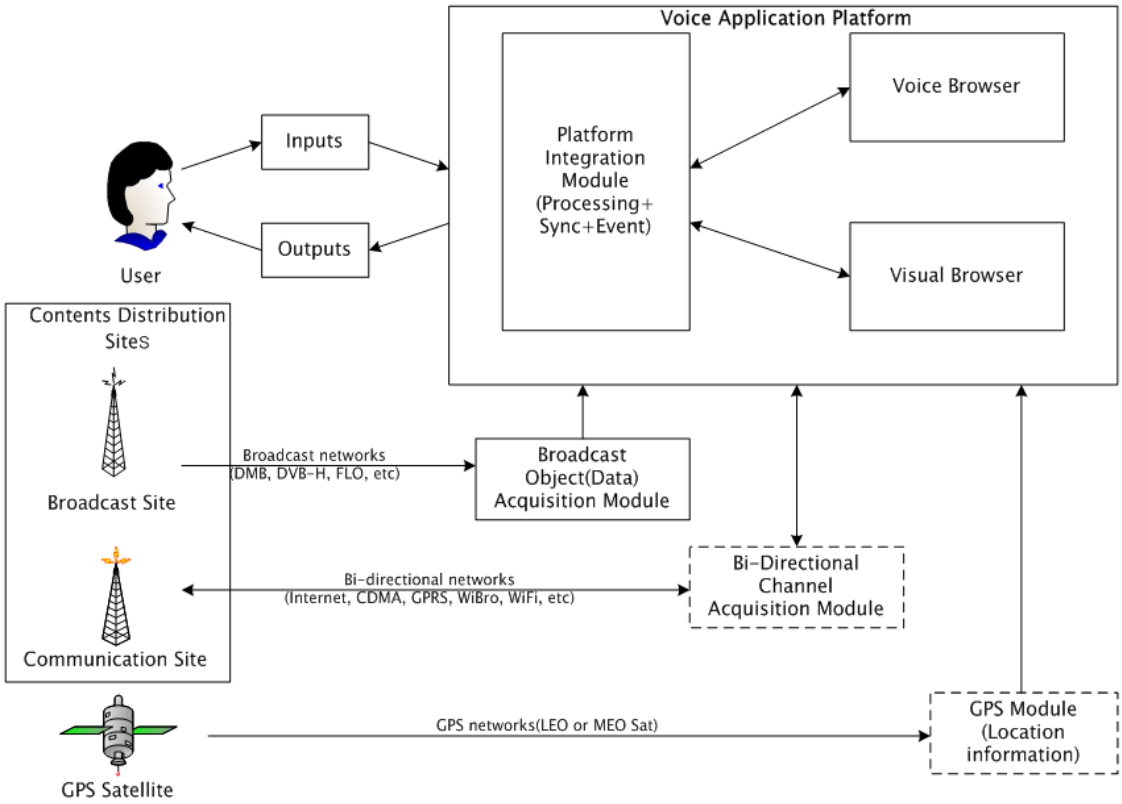



**Figure 1: The Architecture of VA Service Framework**

### 5.1.1 User

As a main player in this application, the user is in the multimodal status when the session is initiated. Then, the user can interact with the VA platform either through the speech interface or the normal graphic interface in order to get access to and eventually navigate through the information. The flexibility provided by such multimodal systems enhances the user's experience considerably and this represents the key benefit brought by these new types of applications.

With regard to user preferences, it is possible to select one of the operation modes: voice only, BWS only or dual mode. On the other hand, there are only two types of content: BWS and voice pages. All pages are authored in accordance with their respective protocols. The user can only interact with voice content while voice mode is active. On the other hand, both content types are available while dual mode is active. Sometimes, the user can change the operation mode by uttering a certain registered keyword or pressing a button that is mounted on the input device.

### 5.1.2 Broadcast site

This actor is also a key player in the VA application domain. It can only be a broadcaster or a mix of content/service provider and multiplexer. The role of the broadcast site is to produce the VA content and to disseminate it. This actor consists normally of several sub-systems ranging from media encoder to several transmission sub-systems. For the VA application, at least one content distribution sub-system must be located at the broadcast site. Sometimes, both distribution sub-systems are linked when both players are involved in VA interactive services by connecting the bi-directional channel to the broadcast network.

In terms of the content delivery, the content distribution sub-system normally transmits the relevant VA data (namely of directory and files objects) using the MOT carousel protocol. The MOT carousel protocol is mandatory for the transport of VA applications.

### 5.1.3 Telecommunication site

As an optional actor, the telecommunication site offers interactive VA content that is not available through the broadcast network (for example due to the shortage of bandwidth capacity). This service channel can be based on a pay model and can offer rich and enhanced content through the bi-directional channel.

In order to provide interactive services, the telecommunication site must not only be able to distribute interactive content but also to support user authentication and billing when needed. Unlike the content distribution sub-system of broadcast sites, this distribution server manages bi-directional service sessions for authentication.

In addition to providing content authored in compliance with the VA protocol, the telecommunication site must ensure that the interactive content is logically linked to VA content transmitted by the broadcast site.

### 5.1.4 Voice Applications platform

This platform is the heart of the VA service chain because it is where the service session takes place. It renders the content and offers different mechanisms to interact with it. It shall be implemented fully in compliance with the VA specification. Basically, it includes the platform integration module and the visual and voice agent engines. These agent engines are also referred to as browsers. The visual browser is a BWS browser. Similarly, the voice browser is an agent that executes the voice content produced in accordance with the VA specification.

The platform integration module can be seen as an operation and management module that controls the whole VA execution. The main role of this module is to operate and manage both agent engines. More specifically, it deals with the synchronous behaviours of both browsers and handles a variety of events. Besides, this integration module also takes care of the operation mode management according to the user's command or the system proprietary policy.

### 5.1.5 Broadcast channel data acquisition module

This module retrieves the original VA content transmitted through the broadcast channel. Normally, the content is reconstructed by the decoding process of MOT carrousel data. In DAB delivery, MOT decoding and suitable binding mechanisms have to be supported.

### 5.1.6 Bi-directional channel data acquisition module

Similarly to the broadcast channel data acquisition module, the bi-directional channel data acquisition module plays a role in retrieving the VA content on the bi-directional channel. And since the telecommunication site is optional, this module is also optional. This module shall reconstruct the VA content and bind the content to the VA platform.

### 5.1.7 GPS module

The GPS module, an optional actor, is provided to support location based services. The location information can be acquired from a GPS receiver which can be located either internally or externally to this platform. When location related content is provided then the GPS module will perform specific operations such as location matching and signalling to the VA platform.

## 5.2 The content structure and delivery

VA content can be made of two types of documents: the voice documents and the sync management documents, see figure 2. Normally, each consists of more than one document. The voice document is a mixture of VoiceXML and some additional elements such as the DAB and the GPS interface elements that adapt and enhance the typical VoiceXML functionality for the context of DAB. The sync document is a standalone document that only contains external sync elements separated from the voice document. The BWS document is entirely isolated from both VoiceXML and sync documents.

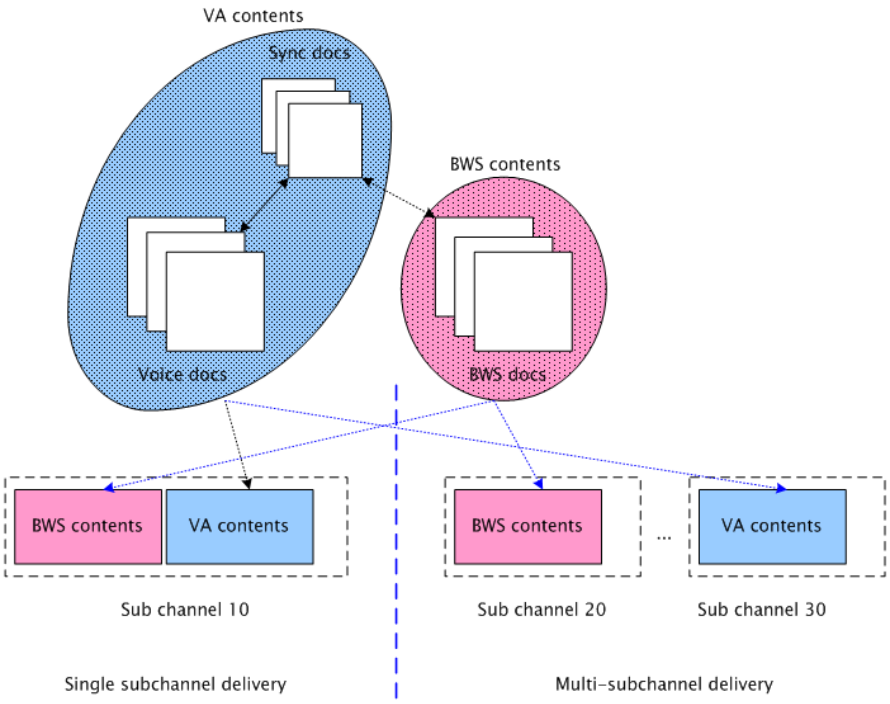



**Figure 2: VA content structure and delivery**

In the present document, voice content (or voice module) is defined as a combination of voice and sync documents while BWS content solely includes HTML documents that comply fully with the BWS specification. Thus, the delivery of both voice and BWS content needs to be managed separately.

In the distribution scenario of VA, BWS content is transmitted and signalled in an isolated way complying with the BWS service. The delivery of VA content shall be taken separately in order to ensure the backward compatibility of BWS application. Also, VA signalling shall be managed in an isolated manner. Accordingly, three operations modes can be defined: voice, BWS, and dual mode.

More specifically, both VA and BWS content can be delivered through separate sub-channels, or sometimes also within the same sub-channel. The container protocol shall be MOT [12]. Eventually, the VA enabled receiver shall be able to retrieve and decode both types of decoupled content. In order to achieve this, it is recommended that the receiver has the capability to decode at least two sub-channels simultaneously.

This delivery scheme guarantees that a receiver with BWS decoding functionality only can execute the BWS application without any consequences from the VA content. On the other hand, advanced receivers with VA capability are still able to support the execution of all operation modes.

The signalling shall be made separately according to its own UAT (User Application Type) and its User application field definition as defined in clause 11.

## 5.3 Namespace

In order for VA to support the sync management and more extended functionalities such as the DAB interface and the GPS interface, it is required to define a new namespace for this purpose.

VA is based on XML and has the following namespace:

< ?xml version= "1.0" encoding = "UTF-8" ?>

The host language of VA is VoiceXML. For the extended elements and attributes, VA shall use the namespace <va:abc>. Herein, "abc" indicates an element or an attribute being extended from VoiceXML.

Thus, the host namespace of VA is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<VoiceXML version="2.0"    xmlns="http://www.w3.org/2001/VoiceXML"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:va="http://www.worlddab.org/schemas/va"
            xsi:schemaLocation= "http://www.w3.org/2001/VoiceXML
                                 http://www.w3.org/TR/voicexml20/VoiceXML.xsd
                                 http://www.worlddab.org/schemas/va
                                 http://www.worlddab.org/schemas/va/va.xsd">
```

## 5.4 Execution model and operation mode

The VA execution model can be represented as depicted in figure 3. The user can interact with both the VA and the BWS browsers. The VA platform can be seen as an extended voice browser that handles the execution of VA content which in turn is made of a combination of VoiceXML and extended elements. The voice browser is for the VoiceXML application. The BWS browser follows the already standardized BWS specification. The sync manager is a key module that handles the synchronization management of both voice and BWS browsers. The sync manager is integrated together with the voice browser to constitute the VA platform.

The arrows indicate the command and data flow to carry out the sync handling of both modalities.

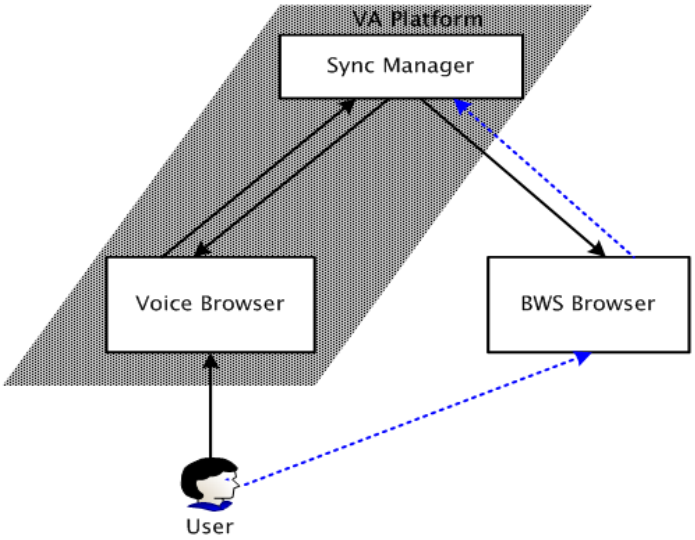



**Figure 3: The execution model of VA**

The VA Platform can handle the user selection for the three operation modes: voice, BWS and dual mode, see figure 4. The voice mode only allows voice content to be executed. On the other hand, the BWS mode is for the BWS exclusive execution. Finally, the dual mode enables the mixed execution of both BWS and voice content in accordance with the sync protocols.

This tri-state mode change is only available while the operation remains within the VA platform. This implies that the user can command a transition from one mode to the other. For example, if a user wishes a dual mode operation, it should be possible when all the input and output modalities are available. Therefore, the system shall support manual or automatic mode switching.

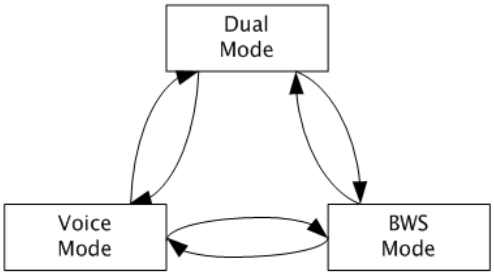



**Figure 4: The tri-state execution mode**

More specifically, in the case of the voice mode, the VA platform is required to discard the interpretation of sync elements. This means that only VoiceXML and other extended elements (i.e. DAB access and GPS associated) are to be interpreted. Sometimes, the extended elements are not supported by the platform. That is the case when the GPS module is not available, for example. Then, the GPS related elements can be ignored by the implemented platform reliably.

Consequently, the tri-state mode can be summarized as follows:

- Voice mode: the BWS browser and sync management is not allowed; sometimes some extended elements can be ignored.
- BWS mode: only the BWS browser can be activated.
- Dual mode: the whole modules are to be enabled; some extended elements can be ignored for specific options.

# 6 Voice Applications basic profile specification

The profile specified in the following clauses defines the basic profile for the Voice Applications user application. This profile is for embedded receivers that are limited in performance and resources.

VoiceXML 2.0 [3] defines the full functionality required to address the creation of audio dialogs that feature synthesized speech, audio recognition of spoken and DTMF inputs, recording of spoken inputs, telephony and mixed initiative conversations. Since VoiceXML 2.0 is originally designed for rich interactive voice response applications, it needs to be modularized so that a simplified platform can be defined for the DAB environment.

In this basic profile, irrelevant functionalities and associated elements are identified and eliminated. For example, telephony and recording related functionality can be discarded. On the other hand, the VoiceXML platform will have to be enhanced with new functionalities which are specific to the DAB context. This is needed to enable the application to get access to the DAB resources, for instance, the Electronic Programme Guide. This extension implies the precise definition of DAB associated elements for the access to these resources.

The size of enhanced VA implementations strongly depend on components such as ASR and TTS. Consequently, the present document also deals with the profiling of those components. Precisely, the speech keywords, DTMF buttons, and speech output are to be addressed for this purpose.

The next clause presents the modularization of VoiceXML 2.0 as well as its profiling for VA. Next, some specifications about the required VoiceXML modules will be presented.

This basic profile includes the sync management, DAB interface and GPS interface as described in the following clauses.

## 6.1 Modularization of VoiceXML 2.0

The VoiceXML 2.0 specification contains a list of "elements" that have different functions. These elements can be grouped into modules to facilitate the integration of VoiceXML 2.0 and Voice Applications. These modules do not change the VoiceXML 2.0 language as specified by W3C. This clause gives a high-level overview of the modules that have been defined in [4].

Voice Applications will only require a subset of the available VoiceXML modules. Each required module will only include the elements that are needed for VA. The unnecessary elements will be discarded.

The difference between the original and the profiled VoiceXML mostly comes from the fact that the DAB environment is fundamentally based on local interaction scenarios where the user will experience information through the speech interface. As unidirectional broadcasting data services do, the local interaction indicates that all the components being involved into the voice applications are to be located at the DAB receiver side.

Table 1 presents an overview of all VoiceXML 2.0 modules. The last column labelled "VA Supportability" summarizes the actual profiling of VoiceXML 2.0 that is included in Voice Applications. The possible values in the column are Y (Supported), N (Not supported) and P (Partially supported).

**Table 1: VoiceXML Modules**

| Module | Purpose | Elements | VA Supportability |
|---|---|---|---|
| Attributes | Common attributes used in VoiceXML | NA | Y |
| Datatypes | Common datatypes used in VoiceXML | NA | Y |
| Document Model | Defines content model for VoiceXML elements | NA | Y |
| Root | VoiceXML stand-alone documents | <VoiceXML> <meta> <metadata> | Y<br><va> is defined for elements and attributes extended from VoiceXML. |
| Forms | Encapsulate voice dialogs | <form> <field> <record> <subdialog> <block> <initial> | P<br>The <record> element is not supported. |
| Menus | VoiceXML menus | <menu> <choice> | Y |
| Enumerate | Enumerate choices or options available to user | <enumerate> | Y |
| Filled | Voice handlers invoked when a slot is filled. | <filled> | Y |
| Events | Events thrown by VoiceXML processor | <catch> <help> <noinput> <nomatch> <error> <throw> | Y |
| Executable statements | Statements for use in voice handlers | <assign> <clear> <var> <log> <reprompt> | Y<br>The <log> element can be ignored by the receiver. |
| Flow control | Flow control constructs from VoiceXML | <if> <else> <elseif> <return> | Y |
| User Input | Speech input constructs from VoiceXML | <grammar> <lexicon> <example> <tag> <token> <item> <meta> <metadata> <one-of> <rule> <ruleref> | Y<br>The only supported grammar is the XML form of the W3C Speech Recognition Grammar Specification (SRGS) [6]. |
| Output | Speech and audio output | <prompt> <value> <audio> <desc> <emphasis> <lexicon> <mark> <voice> <break> <prosody> <say-as> <sub> <phoneme> <p> <s> <meta> <metadata> | Y<br><prompt>, <value> and <audio> are mandatory. The advanced SSML speech elements can be ignored by the receiver. |
| Option | Specify option in a field | <option> | Y |
| Miscellaneous | Non-local transfers in VoiceXML | <exit> <goto> <link> <script> <submit> | Y |
| Resources | Specifying resources for VoiceXML | <param> <property> | Y |
| Object | Foreign objects for VoiceXML | <object> | N |
| Telephony | Telephony control | <transfer> <disconnect> | N |

## 6.2 Voice Applications Modules

### 6.2.1 Attributes

The attributes shall fully comply with VoiceXML [3].

### 6.2.2 Datatypes

The datatypes shall fully comply with VoiceXML [3].

### 6.2.3 Document Model

The document model shall fully comply with VoiceXML [3], except that Voice Applications shall allow the sync documents (esync file) in order to support the synchronization.

### 6.2.4 Root

The metadata information shall fully comply with VoiceXML [3].

### 6.2.5 Forms

The following constraints are imposed on the form creation:

- The <record> form items are not supported.

The remaining features of form creation shall fully comply with VoiceXML [3].

In order to offer directed conversations to the user, the mixed initiative forms shall be supported. However, the functionality of semantic interpretation described in clause 5.2.12 of [3] is not allowed for the MIFs (Mixed Initiative Forms).

### 6.2.6 Menus

Menus shall fully comply with VoiceXML [3].

### 6.2.7 Enumerate

The <enumerate> element shall fully comply with VoiceXML [3].

### 6.2.8 Filled

The <filled> element shall fully comply with VoiceXML [3].

### 6.2.9 Events

The whole features of event handling shall fully comply with VoiceXML [3].

### 6.2.10 Executable Statements

The <log> element can be ignored by the interpreter.

The remaining features of executable statements shall fully comply with VoiceXML [3].

### 6.2.11 Flow Control

The whole features of flow control shall fully comply with VoiceXML [3].

### 6.2.12 User Input

For speech grammars, Voice Applications platform shall support only one common format, the XML Form of the W3C Speech Recognition Grammar Specification [6], for both inline and external grammars. This way, Voice Applications document will have a common XML base language. The support for other grammar formats is disabled in VA.

The platform is not required to be multi-lingual since a mechanism to manage unsupported language or language list is present in SRGS [6]. However, the multi-lingual recognition might be required for some regions.

The Semantic Interpretation for Speech Recognition (SISR) [8] specification is not supported in the VA basic profile. Consequently, the semantic interpretation described in clause 3.1.5 of [3] is not supported. Clause 3.1.6 of [3] describes how to handle recognitions in the case where no semantic interpretation results are available.

It is not mandatory for the receiver to provide DTMF inputs support. However, some new DTMF equivalent keywords can be specified to extend the non vocal control of the system. These new keywords are added to a new VA mode of input. An example of input extensions and the related grammar can be found in annex A. This example is based on a generic cellular phone set of keys.

### 6.2.13 Output

For audio prompting, uncompressed wave files are not supported. Instead, audio objects shall be encoded in MPEG1 (ISO/IEC 11172-3 [9]) or MPEG2 (ISO/IEC 13818-3 [10]) Layer 3 format commonly known as MP3 format. MP3 is a better format for the mobile and portable target platform. Generally, 8 kHz 8 bit mono MP3 files encoded at a bit rate between 8 kbps and 16k bps should offer relatively good results for this application.

There are no restrictions on the data rate, sampling frequencies or submodes.

The file extension in the MOT parameter ContentName shall be "mp3".

Note that the supported audio compression formats are compliant with the TopNews specification [11]. This is to offer consistency in the supported audio codec and maximize the efficiency of delivery.

Here is an example of audio prompting codes using an "mp3" file.

```
<prompt>
    <audio src="welcome_to_WorldDAB.mp3">
        Welcome to the WorldDAB ensemble, we are providing rich multimedia services.
    </audio>
</prompt>
```

With regard to the speech markups that are defined in SSML [7], the following elements are not mandatory and do not have to be supported by the receiver:

- <desc> <emphasis> <lexicon> <mark> <voice> <break> <prosody> <say-as> <sub> <phoneme> <p> <s> <meta> <metadata>.

Only the very fundamental elements such as <prompt>, <audio> and <value> are mandatory for this profile. Therefore, if any of the other elements are encountered, the platform can ignore them.

When some multi-language text is found in a <prompt> element such as in the example below that has the French word "Bienvenue", this enhanced platform must comply with the SSML as a VoiceXML platform does. This means that, while this requires the platform to process documents with one or more "xml:lang" attributes defined, it does not require the platform to be multi-lingual. When an unsupported language is encountered, the platform can ignore the unsupported text or prompt it in an appropriate way, if the TTS is capable of the multi-lingual output.

```
<prompt>
    Welcome, and "Bienvenue" to the WorldDAB ensemble, we are providing rich multimedia services.
</prompt>
```

### 6.2.14 Option

The <option> element shall fully comply with VoiceXML [3].

### 6.2.15 Miscellaneous

The <script> element is supported only for client-side scripting. That is, server-side scripting is not allowed. In addition, the scripting shall be fully compliant to ECMAScript [5].

The remaining features of miscellaneous shall fully comply with VoiceXML [3].

### 6.2.16 Resources

The resources shall fully comply with VoiceXML [3].

### 6.2.17 Object

The object module is not supported in VA. Thus the <object> element is not supported.

### 6.2.18 Telephony

The telephony module is not supported in VA. Thus the <transfer> and <disconnect> elements are not supported.

# 7 Sync Management

This clause describes the mechanisms provided by the VA platform to synchronize speech and graphical outputs as well as various input interfaces such as speech recognition, buttons, stylus, GUI, and so on. VA documents, including those based on the VoiceXML form element, can be synchronized with BWS document. Overall, this will allow the support of the desired multimodal interaction with new applications.

## 7.1 Sync Protocol

Two protocol levels are defined for synchronization:

- Session (application) level.
- Document level.

The session level synchronizes both modes when a session is launched or when transitions between applications occur.

The document level synchronization handles document transitions between the two modes within an active session. It allows the synchronization of VoiceXML document or form to BWS document.

Document level synchronization is achieved with two synchronization schemes:

- The *implicit scheme* uses meta documents and meta elements (tags) specified in separated external documents that contains the synchronization information.
- The *explicit* scheme provides synchronization information explicitly within the VA document.

### 7.1.1 Session Level Synchronization

Session level synchronization occurs when a VA session is established. This information shall be offered in the signalling field the FIG 0/13 information as defined in clause 11. This information is associated with the operation mode type and the identifier of the BWS service component.

This action can be considered as top level synchronization since it happens at the launch of an application. Thus, its role is to synchronize the Voice starter document along with the BWS starter document. If the operation mode is signalled as "dual mode" and at the same time there is identifier of BWS service component, the VA starter page shall be synchronized with the designated BWS starter page. The BWS starter page can be identified from the identifier of BWS service component.

### 7.1.2 Document Level Synchronization

There are three types of document level synchronization:

- document to document;
- form to document;
- arbitrary synchronization.

The document to document type of synchronization is fundamental and it happens when a specific document has to be presented synchronously with another one. For example, if a user currently executing a weather application page wants to move to a traffic application page that presents in same application session, then the relevant sync protocols are to be present somewhere within the content itself. This is done externally as described in clause 7.1.2.1.

Another type of document level synchronization has to be performed to synchronize Voice forms and BWS documents. In this case, one Voice document can be synchronized with several BWS documents. This type of synchronization shall be carried out by using the "explicit method" as described in clause 7.1.2.2.

Arbitrary transitions happen when transitions are specified explicitly with the VoiceXML transition element <link>. Normally, the <link> element is designed to offer the global scope types of transitions. Therefore, if the synchronization element is coupled with the <link> element, the document level synchronization can be accomplished explicitly and sometimes globally without using the VoiceXML <form> element.

#### 7.1.2.1 Implicit scheme

The implicit scheme applies when the following conditions are satisfied. The implicit scheme shall satisfy the sync management algorithm specified in clause 7.2:

- An external "Esync" document that exclusively contains sync information shall be present as a separate file with the file extension ".esync".
- Each "Esync" document shall contain more than one sync element identified by <va:esync>.
- The <va:metasync> element should be present in the voice document to provide a reference to the "Esync" document. The name of the "Esync" document and an element identifier shall be referenced in this element as attributes. The <metasync> element shall be present only once in the initial part of each voice document.
- The BWS document shall not contain any sync element in order to maintain backward compatibility. All of the sync related elements and script codes are presented in the voice documents or in external sync documents and shall be conducted on the VA platform.

For this purpose, the <esync>, <metasync> elements are defined as given in tables 2 and 3.

**Table 2: Attributes of <esync> element**

| Attributes | Description |
|---|---|
| id | A unique <esync> identifier. It is referred in the <metasync>. |
| expr | It specifies an expression to evaluate as a URI to transition to. |
| scope | It's a scope of the synchronization; there are "application", "document" and "form" level scopes. The "application" has the scope of through all documents within an application. The "document" and "form" has a scope of the designated one, respectively. |
| starttime | A time interval. If it is designated as "10s", both documents are to be synchronized within 10s. Using this, time-sensed rendering is available. The default time interval is "0s". |
| vadoc | A URI of VA document to be synchronized. It also indicates a VoiceXML form (i.e. "./ensemble.vxml#sndform"). |
| bwsdoc | A URL of BWS document to be synchronized. |

**Table 3: Attributes of <metasync> element**

| Attribute | Description |
|---|---|
| doc | An URL indicating a document that contains external sync elements. The file name shall have the ".esync" extension. |
| esyncid | The identifier of a specific <esync> element in the Esync document. |

##### 7.1.2.1.1 The placement of the <metasync> element

The <metasync> element can exist in both root and leaf documents. Any <metasync> element in the root document shall have the application scope. The <metasync> element in the leaf document has the scope of the corresponding leaf document. The <metasync> in the root document shall only be used to designate the sync information having the application scope. For this purpose, the "doc" attribute shall be used only in order to indicate the external sync document and the VoiceXML form. Eventually, the designated "esync" document shall be handled as a global document analogous to the root document.

#### 7.1.2.1.2 VA initiated synchronization

The synchronization from the VA entity shall be conducted according to the following steps:

- First, the <metasync> element shall be interpreted in order to collect the meta information when the voice document is loaded.
- The next step is to identify the "Esync" document by referring the <metasync> element. If that is the case, the referenced "Esync" document shall be interpreted.
- The required sync information shall be retrieved from the referred <esync> element specified by the "esyncid" attribute of the <metasync> element. This process leads the VA platform to synchronize both BWS and voice document in accordance with the designated sync protocol.

The process to gather sync information is initiated by interpreting the document from the <metasync> element up to the <esync> element successively. It is recommended that the whole external sync information be managed as global data in order to accelerate the synchronization process and to provide a more effective management.

Figure 5 shows an example of <metasync> element specifying the URL of an "esync" document as well as the identifier of <esync> element. It refers not only to the "travel booking.esync" document, but also to the ID of a specific <esync> element. The sync information such as "starttime" and "URLs" are obtained from the designated "travel_booking" <esync> element.

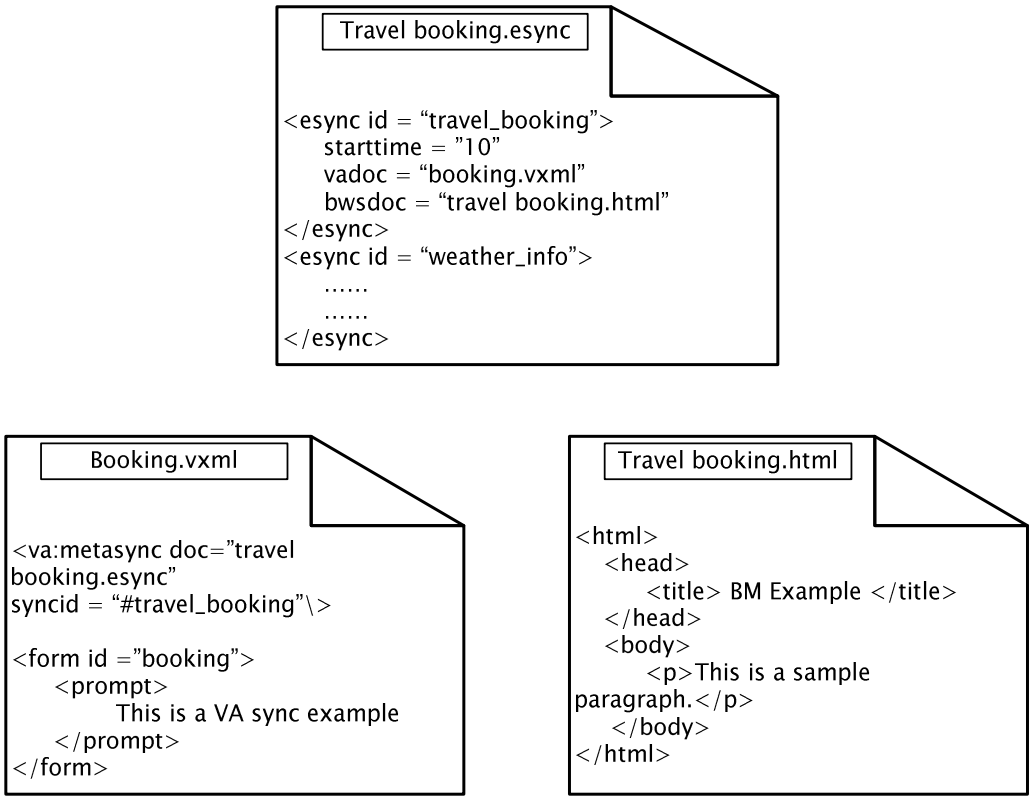



**Figure 5: An example of implicit synchronization scheme**

#### 7.1.2.1.3 BWS initiated synchronization

Synchronization requests caused by BWS entity occur when users click on an anchor specified by the <a> element of BWS document. This kind of synchronization can be realized in a platform-specific way without any modification or update of current BWS profiles. This functionality shall be implemented in accordance with the sync management algorithm described in annex B. The basic scheme is as follows:

- A sync request can be initiated by BWS with the help of the standard HTML anchor tag <a>. These tags are monitored and processed by the VA sync manager, a conceptual module that deals with the sync management between VA and BWS entities. The mechanism by which the VA sync manager can handle this process is not covered in the present document since it is a matter of implementation. It is to note that this functionality can be accomplished by the DOM [Document Object Model] API or in any other platform-specific mechanism.

- The VA sync manager has to manage all possible sync information in order to trigger sync events or requests that are initiated by BWS. If triggered, an associated VA document is loaded and rendered concurrently. The implicit method shall be used for this process. The VA platform must have the capability to load the external sync information with a global scope and must provide the necessary processing power required to execute such tasks.

- If no sync information is found when a transition is initiated by BWS then the BWS document transition shall be executed without any error or influence on the execution of VA platform.

**An example of "Esync" document**

```
<?xml version= "1.0" encoding="UTF-8" ?>
<vxml version = "2.0" xmlns= "http://www.w3.org/2001/vxml"
                      xmlns:va ="http://www.worlddab.org/schemas/va"
                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:schemaLocation="http://www.worlddab.org/schemas/va va.xsd">

<va:esync id = "ex1"
          vadoc = "va1.vxml"
          bwsdoc = "bws1.html"/>

<va:esync id = "ex2"
          vadoc = "va2.vxml"
          bwsdoc = "bws2.html"/>
</esync>
```

### 7.1.2.2 Explicit scheme

The explicit scheme refers to the synchronization scheme that takes place when sync elements are explicitly presented in the enclosed VoiceXML form, the <form> or <menu> element. This scheme allows the support of form to document level synchronization and in addition to the arbitrary synchronization. It is obvious that a VA document can be synchronized with more than one BWS documents. This is because most VA documents consist of more than one form which can be synchronized with the corresponding BWS document. This is accomplished by coupling the associated sync element with the VoiceXML form.

In the case where the sync element is coupled with the common transition element such as VoiceXML <link>, it can also perform an arbitrary synchronization. Here, an arbitrary transition implies an instant transition, normally not driven by any form but rather by the <link> element. The <link> element is also used for the application scope transitions. So when a sync element is coupled with the <link> element, it takes the same scope for the synchronization.

For all these purposes, the <isync> element is defined as given in table 4. The mandatory condition is that the <isync> element shall be appeared in the VoiceXML <form> or <menu> element. More specifically, the VoiceXML form shall have a unique <isync> element having the type attribute of "form". This <isync> element specifies an URL of BWS document to be synchronized with this form.

Sometimes, a VoiceXML form can have more than one <isync> element if <isync> elements are having type attribute of "transit". This is the case where <isync> element exists as a children element of the <link> and <goto> elements of VoiceXML. Normally VoiceXML <link> and <goto> elements are the children elements of VoiceXML form.

So, the "form" type <isync> element is used to synchronize the current VoiceXML form with the designated BWS document as soon as the form is encountered. On the other hand, the "transit" type <isync> element is used to synchronize the next VoiceXML form to be transitioned with also the BWS document to be transitioned.

Therefore when VA platform encounters a VoiceXML form having "form" type <isync> element, then the VoiceXML form shall be spontaneously synchronized with the BWS document designated. At that time, the "transit" type <isync> elements shall be invalid for that synchronization event even though several "transit" type <isync> are existed in that form.

The following sample code shows a good example of <isync> mechanism:

```
<form id = "my_example">
 <va:isync
       id ="my_sync"
       type = "form"
       next = "my_bws.html"/>

<field name="my_input">
    <grammar mode="voice" version="1.0" root="command">
      <rule id="command" scope="dialog">
        <one-of>
          <item> one </item>
          <item> two </item>
          <item> three </item>
</one-of>
      </rule>
    </grammar>

    <prompt>
      Say one, two or three to see more depth information.
    </prompt>

    <filled>
      <if cond="my_input  == 'one'">
        <goto next="#first_information">
          <va:isync id="tr1" type="transit" next="../../first_information.html"/>
        </goto>
      </if>
      ……..
    </filled>
      ……..
    </field>
</form>
```

When "my_example" form is encountered, the "my_bws.html" page is rendered concurrently with the prompting of the message "Say one, two or three for the detailed information". This concurrency prolongs until there is a user input. When the user input is matched with any keyword, then the "transit" type <isync> is valid. For example, if the user utters "one" keyword, "tr1" <isync> element is activated to offer the implicit synchronization with the next VoiceXML form "first_information". In this case, the VoiceXML form, "first_information" dialog, and the next BWS document, "first_information.html", will be synchronized in response to the transition. Table 4 shows the <isync> element and its attributes.

The explicit sync process shall be conducted according to the sync management algorithm specified in clause 7.2.

**Table 4: Attributes of <isync> element**

| Attribute | Description |
|---|---|
| id | Unique <isync> identifier. |
| expr | Specify an expression to evaluate as a URI to transition to. |
| next | An URL of the next BWS document to transition to. |
| scope | It's a scope of the synchronization; there are "application", "document" and "form" level scopes. The "application" scope is effective in all documents within an application. The "document" and "form" scope are effective only in one of these components, respectively. |
| starttime | A time interval. If it is designated as "10s", both documents are to be synchronized within 10s. Using this, time-sensed rendering is available. The default time interval is "0s". |
| type | This value is used for the identification of sync types: form sync or transit sync. The value can be "form" or "transit". The default value is "form". If the value is "form", the designated URI shall be synchronized with the current VoiceXML form. Otherwise, it shall only be allowed at the transition. |

**An example of <esync> and <isync> elements**

```
<?xml version= "1.0" encoding = "UTF-8" ?>
<vxml version = "2.0" xmlns= "http://www.w3.org/2001/vxml"
                  xmlns:va="http://www.worlddab.org/schemas/va"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.worlddab.org/schemas/va va.xsd">

<va:metasync doc="main.esync" esyncid="service_main"/>

<form id="service_main_intro">
  <block bargine="false">
     In this data service, you can get various information dedicated to life in general such as local
hot news,
     local transportation, shopping and so on. Are you ready for surfing this service?
  </block>
</form>

<form id="hotnews_intro">
  <va:isync
     id="hotnews_intro_sync"
     type="form"
     next="hotnews_intro.html"/>

  <field name="move_to_news_page">
    <grammar mode="voice" version="1.0" root="command">
      <rule id="command" scope="dialog">
        <one-of>
          <item> news </item>
          <item> local news </item>
          <item> hot news </item>
          <item> headline </item>
        </one-of>
      </rule>
    </grammar>

    <prompt>
      In this service, headline news and breaking news are provided.
      Do you want to move?
    </prompt>

    <filled>
      <if cond="move_to_news_page == 'news'">
        <goto next="#initial_dialog_for_news">
          <va:isync id="tr01" type="transit" next="../../hotnews.html"/>
        </goto>
      </if>
    </filled>
    <catch event="noinput">
      <goto next="#game_intro"/>
    </catch>
  </field>
</form>

<form id="game_intro">
  <va:isync
     id="game_intro_sync"
     type="form"
     next="game_intro.html"/>

  <field name="move_to_game_page">
    <grammar mode="voice" version="1.0" root="command">
      <rule id="command" scope="dialog">
        <one-of>
          <item> game </item>
        </one-of>
      </rule>
    </grammar>

    <prompt>
      In this service, voice quiz and multimodal games are provided.
      Do you want to play?
    </prompt>
    <filled>
      <if cond="move_to_game_page == 'game'">
        <goto next="#dialog_for_games">
          <va:isync id="tr02" type="transit" next="../../games.html"/>
        </goto>
```

```
     </if>
    </filled>
  </field>
</form>
</vxml>
```

When the VA platform loads this example document, the "service_main_intro" dialog is encountered first. At the same time, an associated "html" document is synchronously rendered as indicated in the external sync document.

Substantially, while the "hotnews_intro" dialog is being executed, the "hotnews_intro.html" document is synchronized as indicated. As a next step, if the user utters the grammar "hot news" for the "move_to_news_page" dialog, and if it is matched, then the synchronous transition will be made to the corresponding "hot_news" dialog and the "hotnews.html" page as well. This is an example of form to document level synchronization.

If there is no input for a while during the execution, an automatic transition to the "game_intro" dialog will be triggered by the event process and in turn this process brings the "game_intro.html" to be rendered simultaneously.

**An example of the <isync> in the <menu> element**

The following example illustrates the explicit synchronization. In here, the <isync> appears along with the <menu> element.

```
<?xml version= "1.0" encoding = "UTF-8" ?>
<vxml version="2.0" xmlns= "http://www.w3.org/2001/vxml"
                    xmlns:va="http://www.worlddab.org/schemas/va"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.worlddab.org/schemas/va va.xsd">

<menu id="meu_example" dtmf="true">
  <va:isync
      id = "menu_example_sync"
      type = "form"
      next = "menu_example.html"/>

  <prompt>
    Welcome to local weather forecast center
    <enumerate>
      For <value expr = "_prompt"/>, press <value expr = "_dtmr"/>
    </enumerate>
  </prompt>
  <catch = "help nomatch">
    <prompt>
      Say one of your interested city like Daejeon
    </prompt>
  </catch>
  <choice next = "./va/total.vxml">
    Main
    <va:isync id="tr01" type = "transit" next = "./bws/total.index"/>
  </choice>
  <choice next = "./va/seoul.vxml">
    Seoul
    <va:isync id="tr02" type = "transit" next = "./bws/seoul.html"/>
  </choice>
  <choice next = "./va/daejeon.vxml">
    Daejeon
    <va:isync id="tr03" type = "transit" next = "./bws/daejeon.html"/>
  </choice>
</menu>
```

When the <isync> is used in a <menu> element, it shall only be appeared inside the <choice> element same as it does in the <link> or <goto> elements.

**An example of <isync> in the <link> element**

```
<script>
  <![CDATA[
    function get_vxml_link( )
    {
      val date, time, location;
      val newLinks;
      date = new Data( ) ;
      time = Date.getHours( ) ;
      location = document.getLocation( ) ;
      ……
      return (newLinks)
```

```
    }

    function get_bws_link( )
    {
        . . . .
    }
  ]]>
</script>
<link expr = "get_vxml_link( )">
  <grammar mode = "voice" version = "1.0" root = "root">
    <rule id = "root" scope = "public">
      <one-of>
        <item> Ensemble </item>
      </one-of>
    </rule>
  </grammar>
  <va:isync id="link01" expr="get_bws_links()"/>
</link>
```

Even though the <link> element appears alone, the <isync> can be coupled with the <link> element as shown in the above example. In this example, an arbitrary transition can be made by the support of the script code to evaluate the next location. For this, an attribute "expr" is used.

## 7.2 Sync Management Algorithm

The sync manager (SM) is the conceptual module that deals with sync management. It is operable when VA needs to synchronize VA component such as VoiceXML dialog and document with a BWS document. The SM is only valid during the dual mode. That is, the voice exclusive or BWS exclusive modes do not need the SM module.

Conceptually located in the middle of both VA and BWS entities, the SM takes control of the whole sync operations. The VA platform shall support the following sync management algorithm:

- VA platform shall refer to the FIG 0/13 field to check the operation mode. The SM shall not be concerned with the BWS exclusive application. That is, the SM shall only be initiated with dual mode execution where VA operates in-sync with BWS.
- SM is forbidden in the cases of mode transitions to voice or dual mode from the BWS mode when the application is initially started as a BWS mode.
- The dual mode execution does not impose the constraint that all the VA and BWS forms and documents have to be synchronized. Instead, at least more than one synchronization event can fulfil the condition of dual mode operation.
- During the dual mode execution, it is recommended that the SM needs to be constantly alive for the background monitoring of the sync calls in order to accept and support abrupt dual mode transitions from the either mode (Voice mode or BWS mode).
- The SM shall initially check the global sync information that might be offered in the root document (root.vxml) as the VA platform is initialized. If it exists, that sync information shall be managed as global variables that affect to the overall application documents.
- At the time when the VA starter page is about to be loaded and executed, the SM has to synchronize both starter pages according to the FIG 0/13 field, if provided. If not provided in the FIG 0/13 field, then the SM has to refer to the global information that appears in the root document. Or, provided that <metasync> is present in the upper part of the VA starter page, then the SM shall use this local sync information instead of the global information.
- It is an option that the initial sync information for the starter page is presented in the FIG 0/13 field or inside the VA starter page itself. That is, the initial sync information can exist either in the FIG 0/13 field, in the root document, or in the starter page. A normative requirement is that the information shall certainly be provided at least once in one of these locations.

- When the execution has to transition to the next document amid of regular VA service, the SM shall check the existence of the <metasync> tag which is normatively located in the beginning part of VA documents. If available, that information takes precedence over the global one that might be in the root document. Provided that there is no sync related information, then the VA document shall be executed only. Note that if there is any BWS document currently displayed, then it is recommended to replace it by an appropriate document (i.e. default document) or to deactivate the browser.

- As a next step, when the execution moves to another form presented in the same document according to a sequentially defined transition, the SM shall check the <isync> tag in the form to be transitioned. If it exists, the form has to be prompted synchronously with the coupled BWS document.

- The implicit <isync> element has priority over the local <metasync> and the global <metasync>. VA shall obey this sync priority rule and therefore the contents are to be authored in the way that maximizes this priority rule.

- The explicit sync element <isync> supports two kinds of synchronization identified by its "type" attribute. If the type is "form", that <isync> element is only valid for the synchronization of between the currently encountered VoiceXML form and the designated BWS document. On the other hand, in the case where the type is "transit", then that <isync> element has to be only valid at the time of transition between the next both VoiceXML form and BWS document to be transited.

- In this case of "transit" synchronization, if the next VoiceXML transition is a document that has a <metasync> element, then the explicit <isync> shall take precedence over the <metasync> element. This rule is sometimes very efficient in that the sync can be established promptly without the interpretation of the <metasync> and the related <esync> element.

- When a mode change happens amidst of the dual mode to the voice mode, the SM shall stay active in order to enable the reinstatement to the dual mode. This condition shall also be guaranteed when the mode returns to the dual mode from the BWS mode.

- The sync handling caused by BWS shall be accomplished by using the global sync information presented in the root document. This means that the explicit sync process from VA to BWS is allowable, but the reverse case is not permitted.

- A VA document shall be associated with one BWS document uniquely. If there is any violation of this principle, then it is recommended that the platform performs the operation according to the sync priority rule. The rule is that the local <isync> is prior to the local <metasync>, and the local <metasync> has precedence over the global <metasync>. Therefore, it is obvious that inconsistent sync information specified in both the implicit <isync> and the <metasync> of the document shall be strongly avoided. If it happens, the execution shall be done in accordance with the sync priority rule.

# 8 DAB Interface

For some VA applications, the control and access to the DAB receiver resources could be required to provide functionalities such as receiver tuning, service selection, and so on. This kind of functionalities can be addressed in the context of VA by introducing an interface between the interpreter and the DAB receiver.

To provide this functionality, three new elements are defined : <tuning>, <watch>, and <reservation>. These new elements shall belong to the <va> namespace to distinguish them from the VoiceXML elements. This will enable program and content guides to be produced for DAB ensembles.

A description of the general mechanism is provided here.

First of all, when the user tunes to an ensemble, the VA interpreter commands and transfers relevant ensemble information, such as the ensemble identifier, frequency, and so on, via the DAB interface module as depicted in figure 6. The DAB interface module play a similar role as the TTS and ASR components do. It deals with the event generation and management between the interpreter and receiver resource.

If the user wants to select a service, a specific program or a specific service component, the interpreter needs to have a protocol to send the specific program identifier or sub-channel identifier to the DAB resource. This function is performed through the DAB interface module, for instance by calling the receiver specific APIs.

Eventually, if there is a demand to make a reservation of any program, then the interpreter needs to send the related parameters to the DAB interface module as the above selection procedure does. The detailed functions of those elements are described in the following clauses.

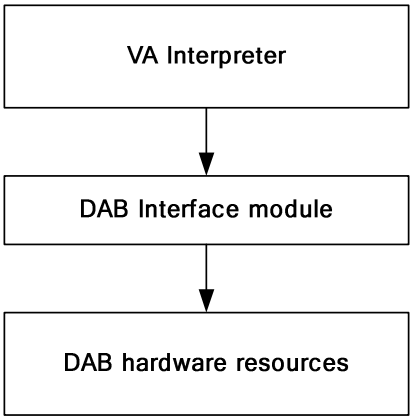



**Figure 6: DAB interface**

## 8.1 <va:tuning>

The <tuning> element is used to tune to a specific ensemble. This <tuning> element shall only appear in the <filled> and <block> elements which are children elements of the VoiceXML <form>. The element and its attributes are defined in given table 5. The "ensembleid" is the value of the ensemble identifier. During the execution of VA, if the interpreter encounters a <tuning> element, the attribute values such as "ensemble id" and "frequency" are to be transferred to the underlying DAB interface module so that it can perform the appropriate tuning operation. In response to it, the DAB interface module controls the actual tuner of the receiver and eventually tunes to the designated ensemble.

**Table 5: <va:tuning> element**

| Attributes | Purpose |
|---|---|
| ensembleid | This attribute represents the unique identifier of the ensemble. The interpreter shall send this identifier to the underlying DAB interface module for the ensemble tuning. This is a string in the form <ECC.EId> in hex format representing the Ensemble Identifier as defined in [1]. |
| frequencytype | This attribute represents the type of ensemble frequency. It can be "primary" or "secondary". |
| frequency | This attribute represents the actual frequency of the ensemble. The unit is kHz. |

## 8.2 <va:watch>

The <watch> element is used to provide a mechanism for the selection of services, programs, or service components. When the interpreter identifies the <watch> element in a VA document, it interacts with the DAB interface module with the associated parameters that are the attributes of <watch> element. The DAB interface module then calls APIs specific to the receiver in order to select a designated service, program or service component in accordance with the transferred parameters from the interpreter.

The <watch> element shall only appear as a children element of the VoiceXML <filled> and <block> elements. As described in table 6, it has "id", "type" and "serviceid" attributes. When a <watch> element representing a service is encountered, the interpreter transfers the <ECC>.<EId>.<SId> value to the underlying DAB interface module to enable the DAB receiver to select the corresponding service. In such case, the value <ECC>.<EId>.<SId> indicates an identifier of one of the DAB services. If the <watch> element represents any service component, then the interpreter shall provide the <ECC>.<EId>.<SId>.<ScIdS> value to the DAB interface module.

**Table 6: <va:watch> element**

| Attributes | Purpose |
|---|---|
| id | An identifier of the watch element. This id can be referred by other elements or by the script. |
| type | This attribute represents one of two possible types of selection: "service", and "service component". It can take the values of "service" and "sc" respectively. |
| serviceid | This represents each identifier of the selected items. Therefore, it has a value of "service id" and "service component id".<br>This is a string in the form <ECC.EId.SId.ScIdS> in hex format representing the service Identifier as defined in the [1]. |

## 8.3 <va:reservation>

The <reservation> element is used to support the reservation of services, programs, or service components being scheduled in the future. When the interpreter encounters the <reservation> element, it interacts with the DAB interface module with the associated parameters that are attributes of <reservation> element. As a response, the DAB interface module performs the reservation operation in accordance with the relevant parameters.

The <reservation> element shall only appear as a child element of the VoiceXML <filled> and <block> elements. As defined in table 7, it has "id", "type" and "serviceid" "time" attributes. When a <reservation> element representing any service (or program) is encountered, the interpreter transfers the <ECC>.<EId>.<SId> value to the underlying DAB interface module to enable the DAB receiver to make a reservation. In the case of a service component, <ECC>.<EId>.<SId>.<ScIdS> shall be transferred.

**Table 7: <va:reservation> element**

| Attributes | Purpose |
|---|---|
| id | An identifier of the reservation element. |
| type | This attribute represents the type of selection, whose possibilities are "service", and "service component". It can take the values of "service" and "sc". |
| serviceid | This represents each identifier of the selected items. Therefore, it has a value of "service id" and "service component id".<br>This is a string in the form <ECC.EId.SId.ScIdS> in hex representing the service Identifier. |
| time | This is a reservation time field in local time. It is based on the ISO 8601 extended format [2]: YYYY-MM-DDThh:mm:ss, where "YYYY" is the year, "MM" the month and "DD" the date. The letter "T" is the date/time separator and "hh", "mm" and "ss" represent the hour, minute and second respectively. To indicate the time zone, i.e. the difference between the local time and UTC, the difference immediately follows the time and consists of a sign, + or -, followed by hh:mm. If this is not present then the difference between local time and UTC is 0. |

Here is an example of DAB interface elements that show the reservation of a program.

```
<form id = "my_radio1">
  <field name = "yes_or_no">
    <grammar type="application/srgs+xml" root="r0" version="1.0"/>
      <rule id ="r0" scope ="form">
        <one-of>
          <item> yes </item>
          <item> No </item>
        </one-of>
      </rule>
    </grammar>
    <block>
      This program is morning news that casts the latest local news along with the weather and
      traffic information for your local area
    </block>

    <prompt>
      Do you want to make a reservation for it? Please say yes or no
    </prompt>
    <filled>
      <if cond="yes_or_no == 'yes'">
        <va:reservatoin id = "prg1"
```

```
                    type = "service"
                    serviceid = "e1.ce10.c111.0"
                    time = "2006-07-01T09:00:00"/>
      </if>
    </filled>
  </field>
</form>
```

# 9 GPS Interface

## 9.1 Location information

Location based applications are enabled in the context of VA when a GPS receiver is embedded into the DAB receiver or else connected to it. The support of position detection within voice applications allows new and innovative location based services to be developed. Standards based geographical coordinates can be used to identify the current location of the terminal and to make the information available to the user as well as to location-based applications. This information will be used for the execution of location dependent content within services.

Location information consists of geographic coordinates. VA applications can use these coordinates to trigger events or to execute special content. Different types of services can use this feature. Table 8 shows some examples.

**Table 8: Example of location services**

| | |
|---|---|
| Location-independent | Most data services such as stock prices, sports reports, etc. |
| Country | Services which are restricted to one country. |
| Regional (up to 200 km) | Weather reports, localized weather warnings, traffic information (pre-trip). |
| District (up to 20 km) | Local news, traffic reports. |
| Up to 1 km | Targeted congestion avoidance advice. |
| 500 m to 1 km | Rural and suburban emergency services, information services (where are?). |
| 75 m to 125 m | Urban SOS, localized advertising, information services (where is the nearest?). |
| 10 m to 50 m | Route guidance, navigation. |

The support of location information in VA is provided by adding one new element and two new types of events. They are described in the following clauses.

## 9.2 <va:location> element

The <va:location> element can be inserted as a children element of a dialog (<form> or <menu>). It specifies the location characteristics that are required in order to trigger the execution of the form items contained within the <va:location> element. The execution of the main loop of the form interpretation algorithm will then determine if some form items have to be visited or not considering the location defined in the <va:location> element. The <registerlocation> element is included in the <va:> namespace.

The <location> element is included in the va namespace. Attributes of <va:location> include.

**Table 9: Attributes of <va:location> element**

| Attribute | Description | Example |
|---|---|---|
| latitude | The latitude of the centre of the described zone. The example can be read as 41d 24.8963' N or 41d 24' 54" N. | 4124.8963 N |
| longitude | The longitude of the centre of the described zone. The example can be read as 81d 51.6838' W or 81d 51' 41" W. | 08151.6838 W |
| altitude | The altitude of the location point. | 1 540 m |
| latituderange | The latitude range around the centre point that is part of the location. | 100 m |
| longituderange | The longitude range around the centre point that is part of the location. | 2 000 m |
| radiusrange | The radius around the centre point that is par of the location. This is mandatory if no latitude and longitude range are defined. | 500 m |
| altituderange | The longitude range around the centre point that is part of the location. | 200 m |

When the <va:location> element is encountered in an application, a validation is made to check if the user's current location is within the range defined by the element. If that is the case then the execution of this part of the VA application will be triggered.

## 9.3 <va:registerlocation> element

This element registers a GPS location into the VA platform so that an event will be generated when that location is reached. The scope of the related event depends on position of the <va:registerlocation> element within the application. The possible scopes are:

- Application: The <va:registerlocation> elements are children of the application root document's <VoiceXML> element. They are initialized when the application root document is loaded. They exist while the application root document is loaded and in any other loaded application leaf document.
- Document: The <va:registerlocation> elements are children of the document's <VoiceXML> element. They are initialized when the document is loaded. They exist only while the document is opened, unless the document is an application root, in which case this is the application scope.
- Dialog: The <va:registerlocation> elements are called in a dialog (<form> or <menu>). The dialog scope exists while the user is visiting that dialog.

The <registerlocation> element is included in the <va:> namespace. Attributes of <va:registerlocation> include.

**Table 10: Attributes of <va:registerlocation> element**

| Attribute | Description | Example |
|---|---|---|
| id | The id of the zone to monitor. The id is returned each time a location event is thrown. | location_12 |
| latitude | The latitude of the centre of the described zone. The example can be read as 41d 24.8963' N or 41d 24' 54" N. | 4124.8963 N |
| longitude | The longitude of the centre of the described zone. The example can be read as 81d 51.6838' W or 81d 51' 41" W. | 08151.6838 W |
| altitude | The altitude of the location point | 1 540 m |
| latituderange | The latitude range around the centre point that is part of the location. | 100 m |
| longituderange | The longitude range around the centre point that is part of the location. | 2 000 m |
| radiusrange | The radius around the centre point that is par of the location. This is mandatory if no latitude and longitude range are defined. | 500 m |
| altituderange | The longitude range around the centre point that is part of the location. | 200 m |

When the <va:registerlocation> element is encountered in an application, the relevant location is registered in the VA platform. The system then monitors the location of the user and generates "va.gps" event according to the registered positions.

## 9.4 va.gps events

Two new events are defined for location based applications. Those two events can be caught as described in clause 5.2.4 of [3]. The special variable "_message" contains the value of the event arguments and it has the structure described in table 11. These events are specific to location information and can be used only with a location reporting device such as a GPS receiver. More details about events handling can be found in [3].

The two location events are va.gps.glocked and va.gps.gupdated. The "glocked" event occurs when the terminal enters the defined zone. The "gupdated" event occurs when the terminal exits the defined zone.

**Table 11: Catch handlers defined in VA location interface**

| Event Type | Message Format |
|---|---|
| va.gps.glocked | Identifier of the corresponding <va:registerlocation> element.<br>Ex.: "location_12". |
| va.gps.gupdated | Identifier of the corresponding <va:registerlocation> element.<br>Ex.: "location_12". |

## 9.5 An example of location information usage

The following example shows a block inside a VA document that registers positions will trigger some events based on location information.

```
<VoiceXML version="2.0" xmlns="http://www.w3.org/2001/VoiceXML"
    xmlns:va="http://www.worlddab.org/schemas/va"
    application="../root.vxml">

<vxml>
<var name="restaurant_events" expr="'no'"/>

<va:registerlocation id="loc1" latitude="4124.8963N" longitude="08151.6338W" radiusrange="100m"/>

<catch event="va.gps.glocked">
    <block id = "restaurant_announce">
        <if cond="restaurant_events=='yes'">
        <if cond="_message=='loc1'">
            <prompt>
                You are currently near a very good seafood restaurant !!
                <audio src="http://www.online-ads.example.com/rest.wav"/>
            </prompt>
        </if></if>
    </block>
</catch>

<form id="rest_1">
    <va:location latitude="4124.8963N" longitude="08151.6838W" radiusrange="500m">
        <field name="info_resto">
    <prompt>Would you like to receive information about nearby restaurants?</prompt>
            <grammar type="application/srgs">yes | no</grammar>
        </field>
        <filled>
            <assign name="restaurant_events" expr="info_resto"/>
        </filled>
    </va:location>
    <block>
        <goto next="#rest_2"/>
    </block>
</form>

<form id="rest_2">
    <block>
        Next statement of the application...
    </block>
</form>

</vxml>
```

When the VA platform loads this document, the registration element <va:registerlocation> is processed. The location coordinates are then registered in a table using a platform dependent mechanism. This table will then be continuously monitored during the execution of the application and compared to the actual location to detect the moment when an event must be generated.

The following <catch> element is the block that handles the va.gps.glocked events that can be generated at any time during the execution of the application. In this case, the handler will compare the _message variable of the event to confirm that the event has been generated by the <va:registerlocation> element that has the id equal to "loc1". Then it will carry out the actions included in the handler.

The form with the id equal to "rest_1" is the logical beginning of the execution of the application. The field named "info_resto" is inserted into the <va:location> element so it will be executed only if the receiver is located inside the physical range described by that element at the time of the execution.

Finally, the application resumes the execution. The <va:registerlocation> element of id "loc1" can still trigger some events throughout the rest of the execution of the VA document.

# 10 Transport of VA

## 10.1 MOT based Delivery

MOT in DAB is a core transport protocol for multimedia objects such as directories and files. It is designed to provide interoperability between:

- Different data services and user application types;
- Different receiver device types and targets;
- Equipment from different manufacturers.

Typically, for every user application, an appropriate transport mechanism has to be specified. In DAB, MOT is the standardized mechanism for the transfer of files over packet mode and X-PAD. Figure 7 illustrates an overview of the MOT encoding and decoding processes.

The implementation of the delivery scenario requires that content/service providers process the various types of multimedia content (e.g. picture, audio clips, text files and execution files) in accordance with the MOT specification and inject these into a DAB ensemble multiplexer.

On the terminal side, the MOT decoder processes and decodes the multimedia content of a DAB Ensemble so that it can be presented to the user.

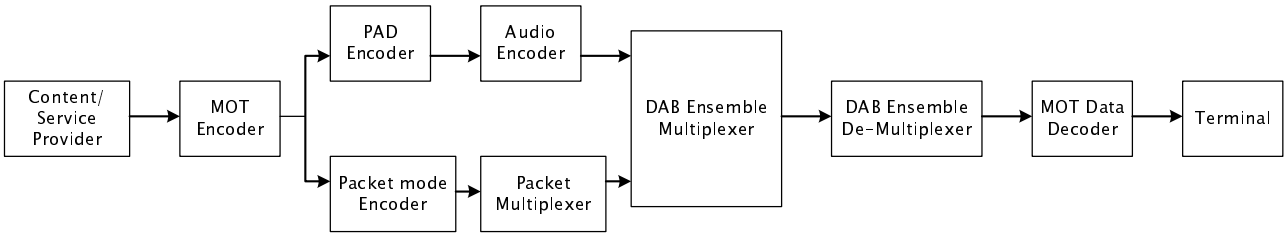



**Figure 7: Overview of MOT encoding and decoding**

As shown in figure 7, all multimedia data services can be delivered either through PAD or packet mode. With PAD, data services can be synchronized with audio services. On the other hand, packet mode content is independent of the audio program.

The main scope of this clause is to adapt the transport modes (PAD or packet mode) and to define key parameters (e.g. contentname, retransmission time and etc) so that VA objects can be delivered via DAB.

### 10.1.1 Transport Mode

Voice Applications may be conveyed in either PAD (X-PAD) or packet mode. In either case, the encapsulation of VA content is fully compliant with the MOT specification.

### 10.1.2 Sub-Channel allocation scheme associated with BWS contents

The allocation of sub-channels for VA content is a key delivery element. When a VA service is independent of audio programs then it has to be transported via packet mode. Content/service providers also have to take into account the sub-channel allocation in conjunction with the BWS content.

Thanks to the flexibility of packet mode multiplexing in DAB, VA content can be multiplexed within a single sub-channel together with BWS content. On the other hand, VA content can be transported separately over a distinct sub-channel. However, the overall efficiency of content acquisition and management at the terminal is optimized when both applications are to be multiplexed into a single sub-channel.

In any of both joint or separate transport scenarios, signalling will allow the terminal to acquire, decode and synchronize the presentation. Signalling is described in clause 11.

### 10.1.3 Terminal behaviour

As a first action, the terminal discovers the existence of specific to be capable of decoding two sub-channels sequentially or simultaneously. Finally, when all objects have been received successfully, the VA platform can load and execute the application.

## 10.2 MOT based Voice Applications

The VA user application type uses the MOT protocol to create a broadcast carousel of files for the voice enabled BWS. As with typical BWS user applications, receivers extract data directly from the carousel in order to present the service. The difference is that the VA will have to use a different service component which shall be described by a different User Application Type. This requirement is mandatory to guarantee that a receiver without a VA decoder will still be able to execute BWS services.

The application required to execute these multimodal websites should be the same for all types of receivers. The platform can however be different on the level of the components that are used to interact with the user. This distinction will be called "implementation platform". The present document proposes two classes of so called implementation platforms. They can be defined as:

- The integrated receiver will contain all the components required to manage multimodal interaction with the user. It will include the TTS and ASR capabilities mandatory to execute all the required tasks. Different levels of TTS and ASR capabilities are possible, so that low-end hardware platforms can use a limited subset of these components. This means that some minimal grammar, as described in annex D, available for ASR and some minimal TTS capabilities will be defined for these integrated platforms.

- PC based receivers will execute the same level of voice enabled BWS but will not have any limits coming from the multimodal interaction components. In fact, this type of receiver will most likely not include any of these components and will enable the use of any of the commercial ASR and TTS products. This feature has the advantage that a PC receiver will be upgradeable as soon as some new and better performing interaction software is available on the market.

These two types of receivers are different only with regard to their interaction capabilities. The VA that will be executing on these platforms will be the same. The integrated platform will depend a lot on its computing capabilities because this determines the quality of the ASR and TTS tools that will be available.

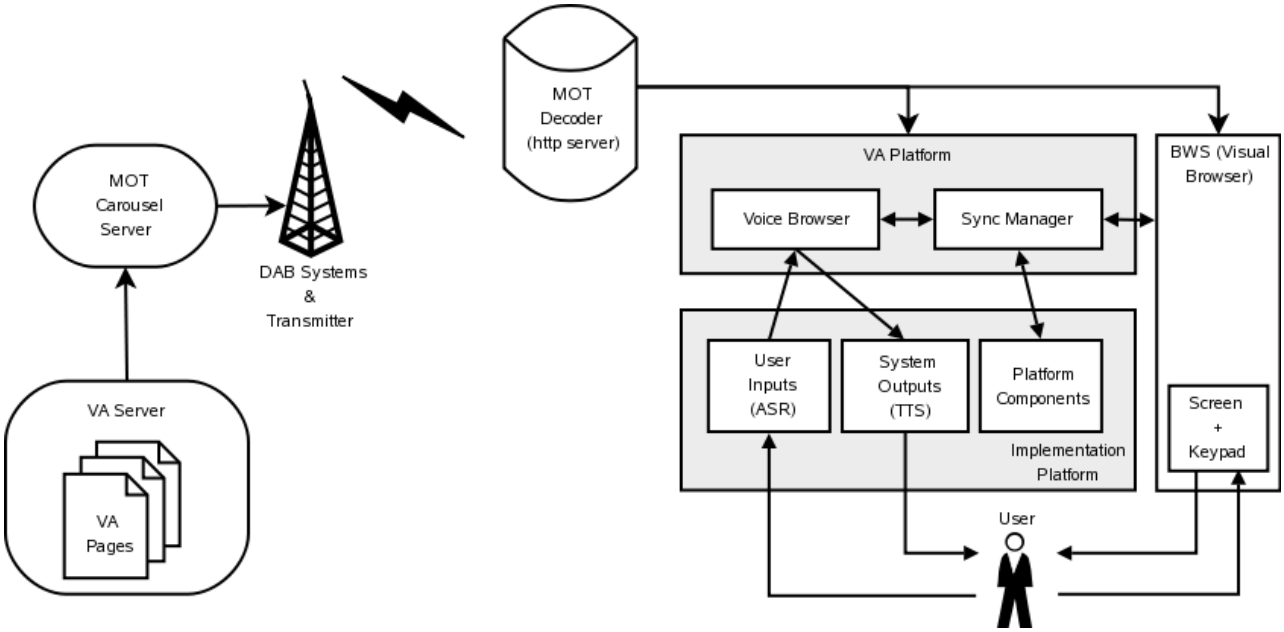



**Figure 8: Operation of the Voice Applications platform**

## 10.2.1 The VA MOT decoder

The VA content has the same nature than a regular website. This means that the pages contain links to other pages within the site. These links are encoded as URLs.

The VA MOT decoder provides a HTTP server interface between the receiver and the VA browser. The VA MOT decoder maps requests of the browser for specific URLs to object within the MOT carousel. The VA browser must communicate through this http interface to assure the full functionality of the provided application.

Whenever an object is requested by the Web Browser, for instance if a link within the currently presented web page is selected, then the VA browser will first translate the probably partial link to its absolute form. It will then send an http request containing the absolute path of the requested object to the VA MOT decoder.

The VA MOT decoder will map the requested absolute path to an object within the MOT carousel with the corresponding ContentName parameter. The object will then be returned to the VA browser in the http response. The object's body will be returned as the http message body and some MOT parameters will be returned as parameters in the http entity header of the response.

The use of http as the communication protocol between Web Browser and BWS Server assures that:

- all MOT parameters can be signalled to the Web Browser;
- the MOT decoder is told which objects are requested at what time by the VA browser. This information is important for cache management;
- if an invalid or yet unavailable object is requested by the multimodal browser, then an appropriate error message can be returned. It is also possible to tell the VA browser to retry loading an object that is not yet available;
- by using the appropriate http parameters, the multimodal browser can be told to automatically recheck for updates of the currently displayed web page at regular intervals.

## 10.2.2 The Voice Applications MOT carousels

Voices Applications use the MOT directory mode as described in [12]. The MOT directory compression is permitted for every carousel containing VA data and for carousels containing data from the BWS advanced profile associated with VA and defined in the present document. The other BWS profiles are defining there own permission for the carousel compression.

The data for the MOT voice enabled broadcast website shall be carried in two MOT carousels, one for the regular BWS content and one for the voice and synchronization content that would interfere with a BWS only decoder. Within each of these MOT carousel, MOT parameters can be associated either with individual objects (by placing them in the MOT header information of an object) or with the entire carousel (by placing them in the MOT directory extension). It will then be possible to set different carousel parameters for the BWS module than the ones for the other modules.

The VA provider shall ensure that the links from one object of any of the MOT data carousels that are part of the application to another object of these MOT data carousels shall be partial (e.g. "../images.jpg" or "/index.vxml"). He shall also assure that no invalid links are contained within the MOT objects (i.e. no links to a file that is not part of the data carousel). The resources that are accessed by the external bidirectional communication link can however result in "resource inaccessible" errors in the case where the external link is not available. The platform shall manage these cases with some specific error messages to the user. As an example, a VA platform that does not have any available interactive channel should instruct the user what he needs to do to have this access (e.g. connect a USB cellular phone or move in range of a WAP). The interactive data that must be accessed using the external bidirectional link is identified in the MOT content as defined in TBD.

Even if service providers can guarantee that only valid links, pointing to actual files, are contained within the carousels, the possibility remains that the browser requests a specific object that is no longer available. This can actually occur during carousel updates .The VA receiver must be able to cope with this eventuality.

### 10.2.3 MOT parameters for individual objects

MOT parameters that are to be applied to individual MOT objects are carried in the MOT header information of the relevant directory entry in the MOT directory. This clause will not cover the MOT parameters for the objects of the BWS Module because they are already defined in [1].

A summary of the use of MOT parameters for individual objects is given in table 12, and is specified in detail by the following clauses. The details regarding all parameters specified in the MOT standard can be found in [12].

The MOT parameters for VA objects are an extension to the parameters defined for BWS objects. This clause will then only define the new parameters and the new specifications of existing parameters. Any parameters that are encountered that are not understood by a given receiver profile shall be ignored.

**Table 12: Use of MOT parameters for individual objects**

| Parameter | Parameter Id | Specified in | Mandatory for UA Provider | Mandatory for Receiver | Occurrence |
|---|---|---|---|---|---|
| ContentName | 0x0C | MOT [12], BWS [1] | Yes | Yes | Single |
| MimeType | 0x10 | MOT [12], BWS [1], present document | Yes | Yes | Single |
| CompressionType | 0x11 | MOT [12], BWS [1] | No | Yes | Single |
| AdditionnalHeader | 0x20 | MOT [12], BWS [1] | No | Yes | Multiple |
| ProfileSubset | 0x21 | MOT [12], BWS [1] | No | No | Single |
| CAInfo | 0x23 | MOT [12], BWS [1] | No | Yes | Single |
| CAReplacementObject | 0x24 | MOT [12], BWS [1] | No | No | Single |
| SubscriberInfo | 0x25 | MOT [12], BWS [1] | No | No | Multiple |

The MOT functionality "conditional access on MOT level" is optional for both VA provider and receiver. However, every MOT decoder shall check if the MOT parameter CAInfo is signalled for a MOT object, thus indicating that the object is scrambled. A receiver with a MOT decoder that does not support conditional access on MOT level shall at least detect and discard scrambled objects.

Every MOT decoder shall check for the MOT CompressionType parameter. Every MOT decoder shall be able to detect and discard MOT bodies it cannot uncompress. In the case of the BWS Module, the BWS profile determines if and which MOT compression schemes the MOT decoder of the receiver shall support.

The MOT carousel used to transport the data for a VA other than the BWS Module may contain objects that have been compressed. The only compression type that is required to be supported by decoders is the "gzip" compression type [14].

#### 10.2.3.1 The ContentName parameter

The use of the ContentName parameter for each object is mandatory. The grammar of the object's name shall conform to the specification of the path component of a fully parsed HTTP URL. The VA decoder shall parse any URL given as part of, for example, a <link> or <goto> tag within a VoiceXML document and use the resultant absolute path to locate the desired object by matching against the ContentName parameter of the objects within the MOT carousel. The matching of the character case shall be exact.

The ContentName shall not use a leading "/". For instance, a ContentName "/images/logo.jpg" is not permitted, whereas "images/logo.jpg" would be permitted.

The ContentName of an object shall not have a trailing "/". For instance, a ContentName "/images/logo.jpg/" is not permitted. This restriction is used to simplify handling of URLs that refer to directories.

ContentNames shall not start with "dgi-bin/", see [1] for details.

The CharSetIndicator field of the ContentName parameter is not significant when matching against a URL path - a byte-by-byte match between the value of the ContentName and the URL path is required. The ContentName parameter is always un-encoded, according to the coding rules specified for URLs.

Sequences of the form "% hex hex" will be treated literally. Table 13 shows some examples of ContentName parameters, together with appropriate references.

**Table 13: Example URL references and corresponding ContentName**

| Path of referencing page | Reference | ContentName |
|---|---|---|
| /a/b/c.html | d.html | a/b/d.html |
| /a/b/c.html | /d/e/f.html | d/e/f.html |
| /a/b/c.html | %2561.html | a/b/%61.html |
| /a/e.html | %65/$/l/f.jpg | a/A/$/l/f.jpg |
| /p/q/r.html | ../s/t.html | p/s/t.html |
| /a/b/c.html | ../../l.html | l.html |
| /a/b/c.html | ./d.html | a/b/d.html |

#### 10.2.3.2 The MimeType parameter

The MimeType parameter indicates the type of an object using the Multi-purpose Internet Mail Extensions (MIME) mechanism (see RFC 2045 [15], RFC 2046 [16], RFC 2047 [17], RFC 2048 [18] and RFC 2049 [19]). MIME strings categorize object types according to a general type followed by a specific format, e.g. text/html, image/jpeg and application/octet-stream.

The basic MIME string may optionally be followed by a ";" and a parameter list. This mechanism is typically used to indicate character sets for text types.

The VA documents are all using XML as their underlying language. The MIME type shall follow the notation from [20]. This RFC document describes the media types related to the W3C Speech Interface Framework. The same methodology has been followed to create a new media type for the esync document, which is a newly defined type of document introduced in the present document.

**Table 14: MIME types related to VA**

| MIME type | Interpretation |
|---|---|
| application/VoiceXML+xml | Voice Extensible Markup Language |
| application/ssml+xml | Speech Synthesis Markup Language |
| application/srgs+xml | Speech Recognition Grammar Specification |
| application/esync+xml | External synchronization |

Table 14 contains the MIME types that are used to identify VA files. The other types that are valid in BWS applications are still available into VA content (e.g. audio/mpeg).

The MimeType parameter is mandatory for every object of the carrousel. The same information can also be found in the ContentType and ContentSubType header parameters of the MOT objects. The MIME protocol is the preferred way to identify types because it is extendable and new types can be added easily. On the opposite, if a new type of object has to be added to the possible ContentType and ContentSubType header parameters, they must be added in the DAB registered tables [2].

Refer to [12] for details on how to set the MOT header parameters ContentType and ContentSubType. For the content that is of the types enumerated in table 14, the MOT object header will have its ContentType and ContentSubType value set to 0 as mentioned previously. These fields should normally be set according to the MIME type of the object. However, as defined in [12], they must be set to 0 (which stands for "general data") when their content type cannot be found in the DAB registered tables [2].

#### 10.2.3.3 The CompressionType parameter

The CompressionType parameter is used to indicate that an object has been compressed and which compression algorithm has been applied to the data. Refer to [2] for the complete list of currently defined compression methods. As stated previously, non compressed data and the "gzip" compression [14] are the only types that are required to be supported by decoders. The VA profile determines if and which other MOT compression schemes the MOT decoder or the user application decoder shall support.

#### 10.2.3.4 The AdditionalHeader parameter

In HTTP, a large number of parameter fields within the http response header may be optionally supplied by a web server in response to a request for a URL. An example of this is the "MIME-VERSION" parameter in the http response header that indicates to a web client, such as a browser, which version of the MIME protocol was used to construct the message.

The AdditionalHeader parameter data field carries an HTTP header field string without any terminating <LF> or <CR><LF> sequence.

#### 10.2.3.5 The ProfileSubset parameter

Where the carousel for a VA service carries objects to support more than one VA profile, additional cache hinting may be applied by the MOT decoder if it knows which profile a given object is used by.

#### 10.2.3.6 Parameters for "Conditional access on MOT level"

The MOT parameter CAInfo is used to indicate the scrambling status of individual objects within the carousel where a service potentially contains both scrambled and unscrambled objects. Even a receiver that does not support conditional access on MOT level shall check this parameter. The existence of this parameter indicates that the MOT object is scrambled. Every receiver shall be able to detect and discard MOT objects it cannot descramble. The support of other MOT parameters used for conditional access on MOT level is recommended for receivers.

The SubscriberInfo Parameter can be used to retrieve details about the required subscriber class for the service.

If a receiver that is compatible with conditional access encounters a scrambled resource to which the user is not registered, it can read the CAReplacementObject parameter in order to get the non protected replacement resource to show instead. This replacement resource can be used to show a message that explains how to proceed to register to the service.

### 10.2.4 MOT parameters for the entire carousel

MOT parameters that are to be applied to the entire carousel are placed in the DirectoryExtension field.

A summary of the usage of MOT parameters for the entire carousel is given in table 15, and is specified in detail by the following clauses. Any parameters that are encountered that are not understood by a given receiver profile shall be ignored.

**Table 15: Use of MOT parameters for the entire carousel**

| Parameter | Parameter Id | Specified in | Mandatory for UA Provider | Mandatory for Receiver | Occurrence |
|---|---|---|---|---|---|
| DirectoryIndex | 0x22 | BWS 1 | Yes | Yes | Multiple |

Sorting of MOT header information (signalled by the MOT parameter SortedHeaderInformation) is recommended for the VA provider.

#### 10.2.4.1 The DirectoryIndex parameter

The DirectoryIndex parameter shall be used as defined in [1].

The DirectoryIndex parameter is used to indicate to the receiver how URLs should be resolved when a URL specifies only a directory. This refers to URLs of the form http://host_name/xyz/ or http://host_name/xyz, where xyz refers to a directory on the web server. Of particular note is the root directory for the service identified by an empty path; this object shall be considered to be the initial object for the service and shall be the first object displayed when the service is started.

On network web servers, this situation is usually handled in one of two ways:

- a configurable default file from within the directory is returned (e.g. index.html);
- an HTML index of the directory is presented.

In the context of the Voice Applications, presenting an index of the directory is not desirable. Instead, the DirectoryIndex parameter may be used to indicate to the receiver the name of the default file name to append to URLs that resolve to a directory within the carousel. In order to support scalable services applicable to a number of receiver profiles, this parameter also specifies the profile of the service for which the given file name should be used.

An empty path ("/") identifies the initial object of a VA. The parameter DirectoryIndex is used to map this path to an object (the initial object) within the root directory. This implies that the initial object(s) (different VA profiles might have different initial objects) must be in the root directory of the VA.

The syntax of the DirectoryIndex parameter data field is given in table 16.

**Table 16: Syntax of the DirectoryIndex parameter data field**

| Syntax | Size | Type |
|---|---|---|
| `DirectoryIndex_parameter_data_field()` | | |
| `{` | | |
| `profile_id` | 8 bits | Unsigned integer, most significant bit first |
| `for (i=0;i<N;i++) {` | | |
| `index_name_byte` | 8 bits | Unsigned integer, most significant bit first |
| `}` | | |
| `}` | | |

**profile_id:** This field identifies the content profile for which the identified object is an appropriate home page.

**index_name_byte:** These fields define the name of the file that should be used as a directory index. No slash ("/") shall be used as part of the index name.

The DirectoryIndex parameter shall be provided for all ProfileIds signalled for the service in FIG 0/13 (see [13]).

# 11 Signalling

## 11.1 User Application Type

VA is defined as a User Application (UA) type within FIG 0/13 (see EN 300 401 [13] and TS 101 756 [2]). The platform profile and application environment parameters are signalled in the user application specific data field as defined in the following clauses.

## 11.2 User Application Specific Data

Following the user application type field, the user application specific data, such as profiling, operation mode and GPS indicators for the signalling of GPS related contents, shall be provided to be conveyed in the FIG 0/13 user application specific data field.

### 11.2.1 Platform Profile Indicator

Voice Applications can have more than one platform and component capability as well. The platform profile indicator helps the receiver determine the VoiceXML level of functionalities as well as the component capabilities. The component capabilities regarding the levels of speech recognition need to be profiled in conjunction with the supported speech keywords.

This version of the document registers the platform profiles as given in table 17. It is a one byte identifier to indicate the level of platform functionalities. "0x01~0x02" is for the basic profile, which is for the resource-constrained embedded receiver as described in clause 6. Specifically, this basic profile is for the receiver that has sufficient performance so that it can support the scripting and relatively high quality of ASR performance. This type of receivers could be a portable laptop, PDAs and car navigation systems. The "basic profile" supports the basic profile of VoiceXML (refer to clause 6) as well as mandatory functionalities of the sync management (refer to clause 7) and DAB interface (refer to clause 8), while the GPS interface (refer to clause 9) is supported as an option.

The quality of user interaction considerably depends on the capability of ASR. ASR is kind of a sensitive element to the reliability of the interaction due to the noise environment. In the case of ASR, a level of vocabularies seems better to be profiled in order to provide choices ranging from the restricted to the unrestricted one. As shown in table 18, the ASR capability indicates the level of vocabulary. The restricted vocabulary restricts the speech vocabularies as specified in annex D. On the other hand, the unrestricted profile has no constraints on the speech keywords to be used.

Each platform profile can be listed up in conjunction with the combination of VoiceXML profile and ASR capability as given in tables 17 and 18.

**Table 17: The registered platform profile**

| Platform Profiles | Description | Explanation |
|---|---|---|
| 0x00 | Reserved | |
| 0x01 | Basic profile with the restricted vocabulary | Refer to "clause 6. VA Basic profile" and annex D for the restricted speech keywords. |
| 0x02 | Basic profile with the unrestricted vocabulary | The unrestricted vocabulary allows the all kinds of two level of keywords, for instance, "headline news". |
| 0x03 ~ 0xFF | Reserved for the future use | |

**Table 18: The registered ASR Capability**

| ASR Capability | Description |
|---|---|
| Restricted vocabulary | The vocabularies of this capability are only supported within the speech keywords classified in annex D. |
| Unrestricted vocabulary | This unrestricted capability embodies full vocabularies can be used for VA. |

### 11.2.2 Operation Mode Indicator

This indicator is to signal the operation mode of VA. From the VA platform perspective, there are two types of modes, "voice mode" and "dual mode". These 4 bits of the operation mode indicator field shall be signalled after the platform profile indicator as given in table 19.

**Table 19: Registered Operation Modes**

| Operation Mode | Description | Explanation |
|---|---|---|
| 0x00 | Reserved | |
| 0x01 | Voice mode | Refer to clause 5.5 and annex C. |
| 0x02 ~ 0x0E | reserved for the future use | |
| 0x0F | Dual mode<br>(multimodal operation with the BWS context) | Refer to clause 5.5 and annex C. In this operation mode, the VA platform can execute only voice contents even though it is signalled as dual mode. It the matter of implementation. |

If the mode identifier is "0x0F", then 12 bits identifier of the service component indicating the BWS contents shall be signalled following the GPS indicator field. If not "0x0F", the 12 bits service component identifier field shall be ignored.

### 11.2.3 GPS Indicator

This indicator is to identify the availability of GPS related contents as given in table 20. "0x0F" indicates that the VA application package contains the GPS related contents and also signals that the VA platform enable the GPS receiver to gather the location information, if available.

As already defined in clause 9 "GPS interface", the VA platform can ignore this field and run without the GPS and its related contents.

**Table 20: GPS indicator**

| GPS Capability | Description | Explanation |
|---|---|---|
| 0x00 | Reserved | |
| 0x01 | No GPS related | VA |
| 0x02~0x0E | reserved for the future use | |
| 0x0F | GPS related | VA |

### 11.2.4 The Syntax of the user application specific data field

The syntax of the user application specific data field is given in table 21.

**Table 21: Syntax of the user application specific data field**

| Syntax | Size | Type |
|---|---|---|
| `User_Application_Specific_Data_Field() {` | | |
| `Platform_profile_indicator` | 8 bits | bslbf |
| `Operation_mode_indicator` | 4 bits | bslbf |
| `GPS_indicator` | 4 bits | bslbf |
| `if(Operation_mode_indicator == 0x0F) {` | | |
| `Service_component_ID` | 12 bits | bslbf |
| `Padding` | 4bits | |
| `}` | | |

**Platform_profile_indicator:** This 8 bits field identifies the content profile for which the identified object is an appropriate VA pages.

**Operation_mode_indicator:** This 4 bits field identifies the operation mode of VA.

**GPS_indicator:** This 4 bits field identifies the existence of GPS related content in the VA package.

**Service_component_ID:** This 12 bits field indicates the service component identifier of BWS as described in clause 6.3 of [1].

In terms of the launching procedure of VA service, first of all, the terminal needs to check the platform_profile_indicator as depicted in the figure 9. If the terminal can afford to cover the platform profile, then the procedure gets into the next step to check the operation mode. Otherwise, the procedure has to be suspended at that point and get sent back to the terminal management. Herein, the terminal management represents a starting point of the VA service.

As a second step, if the mode is indicated as "single mode", then the terminal has to initialize the voice platform exclusively. Otherwise, if it is dual mode, the terminal has to get into the next checking process of the BWS profile as required. After this last check all, if all the profiles turn out to meet with the capabilities of the present platform, then the terminal is supposed to initialize both voice and BWS browsers.

Amid the third step, if the BWS profile is recognized as superior to the built-in BWS platform, for instance the built-in BWS platform is the basic profile while the signalled profile is the unrestricted one, then the single mode operation shall be enabled instead.

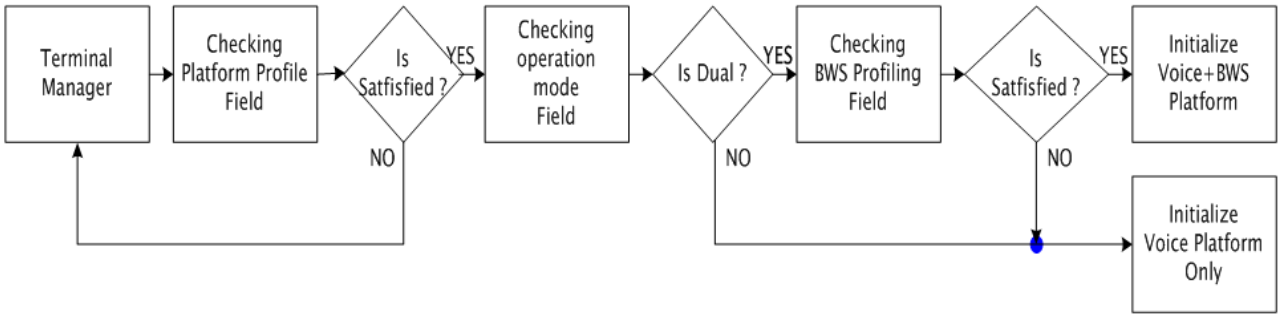



**Figure 9: Application launching process**

# Annex A (normative): User input example

## A.1 Extended key inputs

Some extended key inputs are specified using the common keys from a cellular phone. New keywords such as "Power/End", "Up", "Down", "Right", "Left", "Back/Clr" and "Select" would then be recognized as regular key inputs. Table A.1 shows the keys that are available for the VA system and the type of grammar to which they are associated. Clause A.2 defines the grammar used to represent the extended keys of the VA grammar type.

**Table A.1: Example of a keypad for the Voice Applications platform**

| Keys | Grammar type | Description | Mandatory for receiver |
|---|---|---|---|
| Power/End | -- | This key could be used to shut down the VA browser, however it should not be usable from the application | No |
| Numbers 0-9 | DTMF | Regular numerical keys | No |
| Up, Down, Left, Right | VA | Directional keys | Yes |
| Back/Clr | VA | Cancels the current operation | Yes |
| Select | VA | Accepts or select the currently selected item | Yes |
| Asterisks (*) | DTMF | Asterisk from a numerical keypad | No |
| Pound (#) | DTMF | Pound from a numerical keypad | No |

## A.2 Extended VA grammars

This clause defines the representation of a grammar consisting of some new tokens based on a model of default keys that can be found on portable devices. This grammar is comparable to the DTMF grammar found on a phone except that some keys are different to match the model that is usually found on most portable devices. This VA grammar can be used by a VA platform key inputs system to determine sequences of legal and illegal VA events. All grammar processors that support grammars of mode "va" shall implement this Appendix. However, not all grammar processors are required to support these new VA inputs.

If the grammar mode is declared as "va" then the keywords contained by the grammar are treated as VA input keys (rather than the default of speech tokens).

Each of the VA keys is a legal VA token in a VA grammar. As in speech grammars, tokens shall be separated by white space in a VA grammar. A space-separated sequence of VA keys represents a temporal sequence of these entries. In real systems, VA keys might be used more often to input a 1-key command compared to a complete sequence.

In any VA grammar, any language declaration in a grammar header is ignored and any language attachments to rule expansions are ignored.

In all other respects a VA grammar is syntactically the same as a speech grammar. For example, VA grammars may use rule references, special rules, tags and other specification features.

The following is a simple VA grammar that accepts a PIN composed of 4 consecutive directional keys followed by the "Select" terminator. It also permits the "Back/Clr" key to cancel the operation.

```
<?xml version="1.0"?>
<grammar mode="va" version="1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/06/grammar
        http://www.w3.org/TR/speech-grammar/grammar.xsd"
        xmlns="http://www.w3.org/2001/06/grammar">

<rule id="directions">
<one-of>
    <item> Left </item>
    <item> Right </item>
    <item> Up </item>
    <item> Down </item>
</one-of>
</rule>

<rule id="pin" scope="public">
    <one-of>
        <item>
            <item repeat="4"><ruleref uri="#directions"/></item>
            Select
        </item>
        <item>
            Back/Clr
        </item>
    </one-of>
</rule>

</grammar>
```

# Annex B (normative): Sync Handling

## B.1 The sync handling using global sync information

A global synchronization can be identified that the associated sync information normatively appeared in the root document influences throughout the whole application. The key element of this level of sync handling is to provide the global sync information in the VoiceXML root document as do some other VoiceXML based elements and codes. This is especially to support the sync requests that can be provoked by the BWS part. Eventually, it shall be obeyed that all the sync handlings aroused from the BWS part shall be made by referring this global parameters. In specific, the sync request from BWS part shall not be triggered to the local sync parameters and the global parameters can be used for the sync request from VA to BWS as well.

Thus, a definite factor to meet above features is that at least one <metasync> element shall be offered in the root document as following example:

root.vxml

```
<?xml version= "1.0" encoding = "UTF-8" ?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
                    xmlns:va="http://www.worlddab.org/schemas/va">

<script>
  <!CDATA[
    function myprogrma (x)
    {
      ……
      return (a+b);
    }
  ]]>
</script>

<var name = "current_program" expr="c220"/>
……
<va:metasync doc = "ext1.esync"/>
<va:metasync doc = "ext2.esync"/>
……
</vxml >
```

ext1.esync doc

```
vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml"
                      xmlns:va="http://www.worlddab.org/schemas/va">

<va:esync  id = "ex1"
           starttime = "10"
           vadoc = "vadoc1.vxml"
           bwsdoc = "bwsdoc1.html"/>

<va:esync  id = "ex2"
           vadoc = "vadoc4.vxml#form4"
           bwsdoc = "bwsdoc4.html"/>
 ............
```

In the above example, the referred "ext1.esync" and "ext2.esync" external documents from the "doc" attribute of <metasync> are the additional documents including the specific sync element <esync>, where it defines the document to document and form to document sync protocol. Thus, the sync manager (SM) has to support the sync handling by referring these globally scoped sync parameters when the sync request takes place either from VA or BWS part.

In addition, the "esync" element having the ID of "ex2" designates a VoiceXML form (vadoc4.vxml#form4) to be synchronized along with the BWS "bwsdoc4.html" document. This means that the form to document level of sync information can also be offered in the root document.

# B.2 The sync handling using local sync information

The local synchronization is conducted by making use of both implicit and explicit method. Unlike the global sync information, this local information has a scope that is valid to the loaded document or the encountered form. The local synchronization is applicable to the following cases for:

- the initialization of the starter page;
- the normal execution of VA application.

In order to help the understanding of various facets, some more detail cases are listed in the following clauses.

## B.2.1 The initialization of the starter page

Normally, when the VA starter page is about to be loaded as VA session is established, the VA platform is required to refer the FIG 0/13 UA signalling field to identify the required information such as the BWS location as well as its starter page. If the starter page of the BWS context is designated in the FIG 0/13 UA field, then the VA is able to speed up the sync handling by promptly getting ready for the BWS content and by launching the browser. If not, then the identification of the BWS starter page has to be postponed until the VA root or associated document is interpreted.

The insertion of the locator information into the FIG 0/13 UA field is an informative condition, which means that the starter page sync can be addressed by using the global and local <metasync> element instead. Furthermore, the starter page sync information can be presented in both ways under the strict condition that both informations shall be identical.

This case can be expected when a common transition to the VA starter page takes place during the normal execution of VA. Then, VA platform follows the rule the local <isync>, if it exists, takes precedence over any other <metasync> elements, and therefore these sync elements are prior to the FIG 0/13 information so they shall be obeyed. In a situation where there is no sync element throughout the leaf application documents, then the FIG 0/13 information is recommended to be used.

## B.2.2 An example of local VA execution

The dual mode VA application can be created in a variety of patterns using the sync elements without being fixed to a certain type. This is due to the fact that the way of sync management is flexible as the following example:

Supposing that there is an "example.vxml" document in a VA package, as this document is being interpreted, the SM will scan the <metasync> as a first step for the sync management. If existing, the SM shall process the sync management according to this locally presented <metasync> element even though the same <metasync> element is provided in the root document.

Also if it is assumed that there are five VoiceXML forms in the above example document, no sync tags are affixed to the first to third form in a row, and only form #4 and #5 has its own sync tag, <isync>, then the anticipated execution of this document is that forms #1~#3 have to be synchronized with the BWS document as stated in the <metasync>. After the foreseeable sequential flow, the BWS document can be changed when the interpreter encounters form #4 as well as form #5, respectively.

Besides, a situation can be postulated that a transition to this example document can be made from other VA document and even further the transition triggers the form #3. The response of SM will give the insight into processes that the SM will check the existence of the <isync> tag and examine the global sync information because the <isync> does not exist. If the global sync does not exist either, then the form #3 does not need to be synchronized with the BWS document. This example gives an obvious idea that the local <metasync> element is only valid while the document is loaded and the execution is performed from the first paragraph as well.

## B.3 The cases: mixture of <metasync>, <esync>, and <isync> elements

### B.3.1 No sync elements

This case is of the voice mode VA application, thus the SM does not need to be operable.

### B.3.2 Using only global sync elements

This case contains the meaning that it supports and manages all levels of sync behaviour, such as form to document and document to document in a global scope, where all the sync related information are managed as global parameters. Thereby, if all the sync parameters are sufficiently provided, there is no reason for <isync> elements to be used. This is a very preferable method in the aspect that it can simplify the sync description and management, except that all the sync parameters are stored and handled as a global data, which might have a negative effect on the memory constrained devices.

### B.3.3 Using only local sync elements

This is the case where global sync elements are not used. This scenario can be useful especially for the voice driven application which the only interaction is driven by the VA entity and as a result the BWS has to respond merely triggered to the VA part. Thus, the interaction on the BWS browser has to be disabled in an appropriate way. A voice driven slide show application might be the case of this type of VA application.

### B.3.4 Using both global and local elements

To make the better application, it is recommended to avoid the redundant descriptions of sync elements and to keep the priority rule of <isync>, local <metasync>, and global <metasync>. In order to reduce the global memory and ensure the fast sync behaviours as well, it is also recommended that the <isync> element is used appropriately along with the global elements. A dual mode VA application is quite suitable for this case.

### B.3.5 Miscellaneous

Figure B.1 illustrates a situation where a <metasync> in "abc.vxml" document and a <isync> element in different "cde.vxml" document are referring the same "target.vxml" document. A noticeable feature is the referred "target.vxml" document can be focused either by the reference of <metasync> or <isync> element in different way. Herein, what has to be obeyed is the destination of BWS item shall be identical. In other words, the value of "next" attribute in <isync> and the value of "bwsdoc" attribute in <esync> shall be exactly the same.

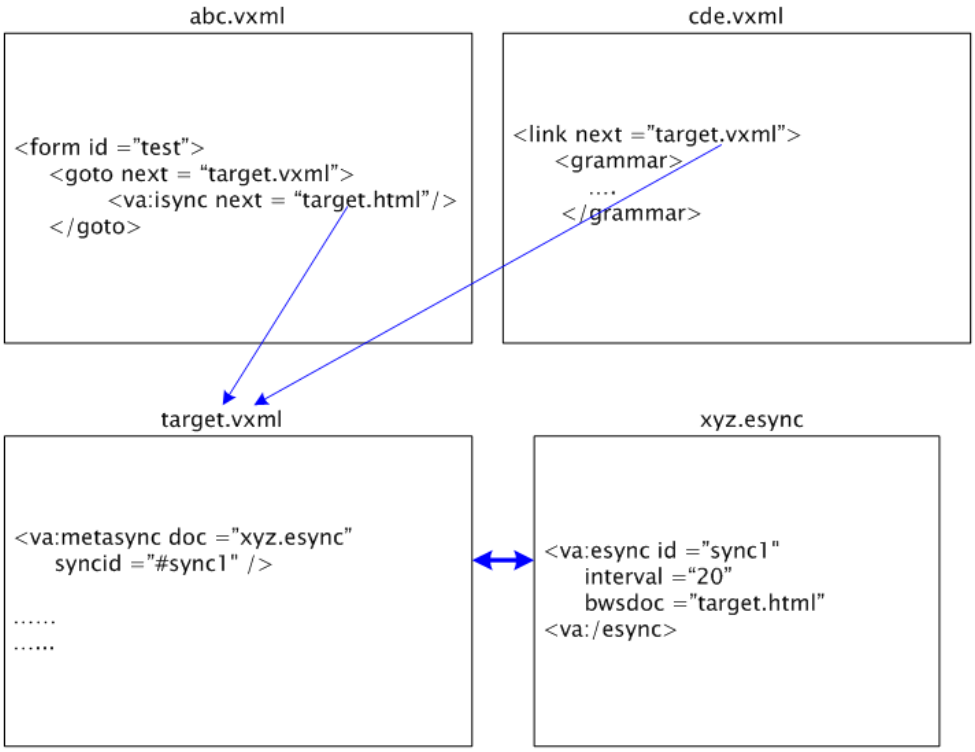



**Figure B.1: An example of simultaneous referencing of the <metasync> and <isync> elements**

## B.4 Sync handlings driven by the BWS entity

It is only applicable in the dual mode operation and it is available when the Sync Manager (SM) takes control of this process as well. As already noted, this type of synchronization shall be accomplished by using only global sync information while the action is triggered to the document transitions requested by the user from the BWS side.

In the case where the relevant sync parameter does not exist while the sync event is provoked in BWS side, it must be satisfied not only that the corresponding BWS document shall be only displayed but also that the SM has to suspend the VA platform with only enabling the voice user inputs. That is, the process of the VA has to be suspended over the previous VoiceXML form and at the same time the user input is only allowed instead, which is to enable the user to command for the transitions via speech inputs (ASR and DTMF) and is to eventually resume the VA platform.

Above case also implies that the out-of-sync BWS contents can be executed for a while until there is an in-sync action. Consequently, the implemented VA platform has to properly support this kind of sync handling driven by the BWS and the arbitrary mode changes as described in the following annex C.

# Annex C (normative): Management of operation mode

With regard to the tri-state execution mode, the user can do switch over three modes under the condition that VA has started with the dual mode. As a matter of implementation issue (that is, an optional function), the VA platform is capable of coping with the following situations, if supported:

- The situation where the screen display is unavailable during the dual mode operation.
- This situation can happen either by the platform itself or especially when the user is on the move. As the former case, the suspension of the BWS browser or the shutting down of the screen as a minimal action can be conducted by the implemented platform itself, for instance by being triggered to the speed limit.
- In addition, the compulsory mode transition, for example from the dual to the voice mode when the user wants to keep watching the TV program while listening to the voice application, can be made by the user. An obvious thing is that the SM execution shall be continued even under the forced uni-mode operation in order for the platform to be able to resume the dual mode or to switch to the intended mode.
- When the speech input is unattainable.
- Sometimes the speech input turns out to be an inappropriate solution under severe noise environment for instance, where the fidelity of speech input goes below the expectation. In this case, the platform can be asked to halt the VA platform as well as to turn off the sound output forcibly either by the platform itself or the user. The execution of SM shall be sustained even in this kind of situation in order to enable mode transitions.
- Virtually, what considerations can be made are not only to facilitate the special speech keyword (i.e. "computer", which is a key to activate the user input during the idle input status) that enable the VA platform, but to utilize the GUI interface (like button on the navigation bar) that allow the user to interact for the mode change as well.
- Mode transition from the Voice mode to the BWS mode or vice versa.
- It is recommended to support these transitions in a proper way by the implemented platform. Normatively the execution of SM should be guaranteed even under certain circumstances.

# Annex D (informative): Speech Keywords

The speech keyword is kind of recognizable word normally expected to be commanded by the user. In VoiceXML [3], the Grammar deals with the speech keyword that has to be recognized through the ASR engine. VoiceXML Grammar basically covers even the natural understanding, called the semantic interpretation aiming the full interpretation of natural speech. However, since a constraint has imposed on our VA domain that only two sequential words are allowed, the semantic interpretation does not need to be supported anymore.

In terms of the speech recognition, the current level of engines seems insufficient especially in the performance, mostly, for the embedded cases. The quality of the engine is normally proportional to the environment, the capability of noise reduction, and the vocabulary supported, and eventually these features bring down the overall performance. Despite of the gradual growth of voice (speech) application market and the enhancement of algorithm, there is still a gap in fully supporting the whole speech words, called semantic interpretation.

The following guideline of speech keywords will help enhance the ASR capability because DAB shares the characteristics with the common embedded device. This category is the one that is registered as one of ASR Capabilities as described in table 18.

It is recommended that each country not only to replace above keywords into his own language, but also to add more keywords according to their needs.

## D.1 The Objective

Classifying the useful speech keywords is a good way to supplement the above noted drawback by giving an opportunity to tune up the ASR engine, which implies that the ASR at least has to support the recognition of the given classified speech keywords. This is the reason why those keywords are frequently used and sometimes useful keywords in the context of DAB. In other words, if both content providers and platform makers follow this guideline in developing the application and implementing the platform respectively, the reliability of the ASR and furthermore the quality of VA application could be augmented.

The speech keywords can be classified by the following categories:

- Navigation keywords.
- Service keywords.
- Program type keywords.
- Data service keywords.
- Control keywords.

## D.2 Navigation keywords

Navigation keywords address the common speech keywords related to navigation behaviours. Only a couple of essential keywords are listed in here, instead more keywords are still able to be appended according to the proprietary purpose of each county and even each broadcaster.

The following navigation keywords are recommended:

- Main.
- Menu.
- Backward.
- Forward.
- Up.
- Down.
- Search.

## D.3 Service Keywords

Service keywords deal with the name of DAB services registered in [2]. For example, in the case of video service so called DMB, the term "DMB" can be registered as one of service keywords.

The followings are recommended based on the given user application type table 16 specified in [2]:

- Radio service.
- Slideshow.
- Broadcast Website.
- TPEG.
- TMC.
- EPG.
- DAB Java.
- DMB or Video service.
- IPDC.
- Voice Applications.
- Middleware.
- Journaline.

Further application or service types can be added. These keywords can be useful when it is used as a grammar in the actual voice application to designate any transition to the service or application so that the user can directly locate it.

## D.4 Program type Keywords

Program type keywords are to identify the program types specified in [2]. These keywords are also able to be used to identify the genre based audio program especially for the EPG application. In an EPG application written according to the VA protocol, the genre based EPG guide pages can be sorted to present in response to the user utterance of one of the genre keywords. Basically, the general keyword of program types are only permitted as the programme type keywords in the context of VA.

## D.5 Data service Keywords

Data service keywords are to identify the unclassified applications in the DAB UATs (User Application Types). Normally this is for the voice applications. Frankly, a variety of data applications can be made in the type of Voice Applications. In that case, the supporting of unregistered application types enables the direct transition of page to page (or dialog to dialog) by allowing user to command the associated speech keyword. As an example, supposed that a local traffic information page is provided in a VA session and the user wants to move to that specific page, then the best way could be to just simply command the designated keyword "local traffic", for instance.

In some cases, the program type keywords can also be used in this kind of data service category.

This clause covers the additional keywords that are not registered in [2] as follows:

- Traffic.
- Local traffic.
- Stock.
- Advertisement.
- Game.
- Local information.

Future extension can be made available if new data service types are required to be registered.

## D.6 Control Keywords

Control keywords are for the overall control behaviours of the VA platform. For example, the keyword "computer" can be normally used as a high level command to activate the VA platform. By uttering this "computer" keyword, the user can commence or resume the speech interaction with the VA platform exactly by activating or reactivating the ASR engine.

- Computer.
- Terminate.
- Resume.
- Suspend.
- Mode change.

Future extension will be made available if new control keywords are required to be registered.

# Annex E (informative): VA Schema

The schemas for VA are based on the modified version of VoiceXML2.0 [3] to restrict and extend some functionalities. The main schemas are "va.xsd" and "vxml.xsd". "va.xsd" schema is newly defined to support the DAB environment while "vxml.xsd" has some restriction and extension in accordance with the VA specification. The other schemas relevant to VoiceXML, such as vxml-datatypes, vxml-attributes, grammar-core.xsd, synthesis-core.xsd, vxml-grammar-restriction, vxml-grammar-extension, vxml-synthesis-restriction and vxml-synthesis-extension are fully complied with the VoiceXML2.0 [3] ratified by W3C.

## E.1 va.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.worlddab.org/schemas/va"
            xmlns="http://www.worlddab.org/schemas/va"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:vxml="http://www.w3.org/2001/vxml"
            elementFormDefault="qualified" version="1.0">

    <xsd:annotation>
        <xsd:documentation>VoiceXML 2.1 schema (20040713) </xsd:documentation>
    </xsd:annotation>

    <xsd:import namespace="http://www.w3.org/2001/vxml"
                schemaLocation="vxml.xsd"/>

    <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
                schemaLocation="http://www.w3.org/2001/xml.xsd"/>

    <xsd:simpleType name="va.mode">
        <xsd:annotation>
            <xsd:documentation>va's mode</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:NMTOKENS">
            <xsd:enumeration value="dual"/>
            <xsd:enumeration value="visual"/>
            <xsd:enumeration value="voice"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="va.script.datatype">
        <xsd:annotation>
            <xsd:documentation>Script Expression (ECMA-262 ECMAScript)</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="va.scope.datatype">
        <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="document"/>
            <xsd:enumeration value="application"/>
            <xsd:enumeration value="global"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="va.sync.scope">
        <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="application"/>
            <xsd:enumeration value="document"/>
            <xsd:enumeration value="form"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="va.esync.type">
        <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="application"/>
            <xsd:enumeration value="document"/>
        </xsd:restriction>
    </xsd:simpleType>
```

```
<xsd:simpleType name="va.isync.type">
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="form"/>
        <xsd:enumeration value="transit"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="va.duration">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="(\+)?([0-9]*\.)?[0-9]+(ms|s)"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="va.freqtype">
    <xsd:annotation>
        <xsd:documentation>va's frequency type</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:NMTOKENS">
        <xsd:enumeration value="primary"/>
        <xsd:enumeration value="secondary"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="va.dab.type">
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="service"/>
        <xsd:enumeration value="sc"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:attributeGroup name="va.gps.attribs">
    <xsd:attribute name="latitude" type="xsd:string"/>
    <xsd:attribute name="longitude" type="xsd:string"/>
    <xsd:attribute name="altitude" type="xsd:NMTOKEN"/>
    <xsd:attribute name="latituderange" type="xsd:NMTOKEN"/>
    <xsd:attribute name="longituderange" type="xsd:NMTOKEN"/>
    <xsd:attribute name="radiusrange" type="xsd:NMTOKEN"/>
    <xsd:attribute name="altituderange" type="xsd:NMTOKEN"/>
</xsd:attributeGroup>

<xsd:element name="esync">
    <xsd:complexType>
        <xsd:attribute name="id" type="xsd:ID"/>
        <xsd:attribute name="expr" type="va.script.datatype"/>
        <xsd:attribute name="scope" type="va.sync.scope"/>
        <xsd:attribute name="starttime" type="va.duration"/>
        <xsd:attribute name="type" type="va.esync.type"/>
        <xsd:attribute name="vadoc" type="xsd:anyURI"/>
        <xsd:attribute name="bwsdoc" type="xsd:anyURI"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="metasync">
    <xsd:complexType>
        <xsd:attribute name="doc" type="xsd:anyURI"/>
        <xsd:attribute name="esyncid" type="xsd:ID"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="isync">
    <xsd:complexType>
        <xsd:attribute name="id" type="xsd:ID"/>
        <xsd:attribute name="expr" type="va.script.datatype"/>
        <xsd:attribute name="next" type="xsd:anyURI"/>
        <xsd:attribute name="scope" type="va.sync.scope"/>
        <xsd:attribute name="starttime" type="va.duration"/>
        <xsd:attribute name="type" type="va.isync.type" default="form"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="tuning">
    <xsd:complexType>
        <xsd:attribute name="ensembleid" type="xsd:NMTOKEN"/>
        <xsd:attribute name="frequencytype" type="va.freqtype"/>
        <xsd:attribute name="frequency" type="xsd:NMTOKEN"/>
    </xsd:complexType>
</xsd:element>
```

```
    <xsd:element name="watch">
        <xsd:complexType>
            <xsd:attribute name="id" type="xsd:ID"/>
            <xsd:attribute name="type" type="va.dab.type"/>
            <xsd:attribute name="serviceid" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="reservation">
        <xsd:complexType>
            <xsd:attribute name="id" type="xsd:ID"/>
            <xsd:attribute name="type" type="va.dab.type"/>
            <xsd:attribute name="serviceid" type="xsd:string"/>
            <xsd:attribute name="time" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="location">
        <xsd:complexType>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element ref="vxml:catch"/>
                <xsd:element ref="vxml:help"/>
                <xsd:element ref="vxml:noinput"/>
                <xsd:element ref="vxml:nomatch"/>
                <xsd:element ref="vxml:error"/>
                <xsd:element ref="vxml:filled"/>
                <xsd:element ref="vxml:initial"/>
                <xsd:element ref="vxml:link"/>
                <xsd:element ref="vxml:property"/>
                <xsd:element ref="vxml:script"/>
                <xsd:element ref="vxml:subdialog"/>
                <xsd:element ref="vxml:block"/>
                <xsd:element ref="vxml:data"/>
                <xsd:element ref="vxml:field"/>
                <xsd:element ref="vxml:var"/>
                <xsd:element ref="vxml:grammar"/>
                <xsd:element ref="isync"/>
                <xsd:element ref="registerlocation"/>
            </xsd:choice>
            <xsd:attributeGroup ref="va.gps.attribs"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="registerlocation">
        <xsd:complexType>
            <xsd:attribute name="id" type="xsd:ID"/>
            <xsd:attributeGroup ref="va.gps.attribs"/>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

# E.2 vxml.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.w3.org/2001/vxml"
            xmlns="http://www.w3.org/2001/vxml"
            xmlns:va="http://www.worlddab.org/schemas/va"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
    <xsd:annotation>
        <xsd:documentation>VoiceXML 2.1 schema (20040713) </xsd:documentation>
    </xsd:annotation>
    <xsd:annotation>
        <xsd:documentation>
            Copyright 1998-2004 W3C (MIT, ERCIM, Keio), All Rights Reserved. Permission to
            use, copy, modify and distribute the VoiceXML schema and its accompanying
            documentation for any purpose and without fee is hereby granted in perpetuity,
            provided that the above copyright notice and this paragraph appear in all
            copies.  The copyright holders make no representation about the suitability of
            the schema for any purpose. It is provided "as is" without expressed or
            implied warranty.
        </xsd:documentation>
    </xsd:annotation>
```

```
<xsd:annotation>
    <xsd:documentation>
        Numeric references are to sections in VoiceXML 2.0.
        [REFERENCE] refers to a reference in VoiceXML 2.0.
    </xsd:documentation>
</xsd:annotation>
<xsd:annotation>
    <xsd:documentation>
        Importing dependent schemas including datatypes,
        attributes and adapter schemas for SRGS 1.0  and SSML 1.0
    </xsd:documentation>
</xsd:annotation>

<xsd:include schemaLocation="vxml-datatypes.xsd"/>
<xsd:include schemaLocation="vxml-attribs.xsd"/>
<xsd:include schemaLocation="vxml-grammar-extension.xsd"/>
<xsd:include schemaLocation="vxml-synthesis-extension.xsd"/>

<xsd:import namespace="http://www.worlddab.org/schemas/va"
            schemaLocation="va.xsd"/>

<xsd:import namespace="http://www.w3.org/XML/1998/namespace"
            schemaLocation="http://www.w3.org/2001/xml.xsd"/>

<xsd:annotation>
    <xsd:documentation>Common Content Models</xsd:documentation>
</xsd:annotation>
<xsd:complexType name="basic.event.handler" mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:group ref="executable.content"/>
        <xsd:group ref="dab.content"/>
        <xsd:element ref="va:isync"/>
    </xsd:choice>
    <xsd:attributeGroup ref="EventHandler.attribs"/>
</xsd:complexType>
<xsd:group name="audio">
    <xsd:choice>
       <xsd:element ref="enumerate"/>
       <xsd:element ref="value"/>
       <xsd:element ref="audio"/>
    </xsd:choice>
</xsd:group>
<xsd:group name="input">
    <xsd:annotation>
        <xsd:documentation>input using adapted SRGS grammars</xsd:documentation>
    </xsd:annotation>

    <xsd:choice>
        <xsd:element name="grammar" type="mixed-grammar"/>
    </xsd:choice>

</xsd:group>
<xsd:group name="event.handler">
    <xsd:choice>
        <xsd:element ref="catch"/>
        <xsd:element ref="help"/>
        <xsd:element ref="noinput"/>
        <xsd:element ref="nomatch"/>
        <xsd:element ref="error"/>
    </xsd:choice>
</xsd:group>
<xsd:group name="dab.content">
    <xsd:choice>
        <xsd:element ref="va:tuning"/>
        <xsd:element ref="va:reservation"/>
        <xsd:element ref="va:watch"/>
    </xsd:choice>
</xsd:group>
<xsd:group name="executable.content">
    <xsd:choice>
        <xsd:group ref="audio"/>
        <xsd:element ref="assign"/>
        <xsd:element ref="clear"/>
        <xsd:element ref="data"/>
        <xsd:element ref="exit"/>
        <xsd:element ref="foreach"/>
        <xsd:element ref="goto"/>
        <xsd:element ref="if"/>
```

```
        <xsd:element ref="log"/>
        <xsd:element ref="reprompt"/>
        <xsd:element ref="return"/>
        <xsd:element ref="script"/>
        <xsd:element ref="submit"/>
        <xsd:element ref="throw"/>
        <xsd:element ref="var"/>
        <xsd:element ref="prompt"/>
    </xsd:choice>
</xsd:group>
<xsd:group name="foreach.content">
    <xsd:choice>
        <xsd:group ref="audio"/>
        <xsd:element ref="assign"/>
        <xsd:element ref="clear"/>
        <xsd:element ref="data"/>
        <xsd:element ref="exit"/>
        <xsd:element ref="goto"/>
        <xsd:element ref="if"/>
        <xsd:element ref="log"/>
        <xsd:element ref="reprompt"/>
        <xsd:element ref="return"/>
        <xsd:element ref="script"/>
        <xsd:element ref="submit"/>
        <xsd:element ref="throw"/>
        <xsd:element ref="var"/>
        <xsd:element ref="prompt"/>
    </xsd:choice>
</xsd:group>
<xsd:group name="variable">
    <xsd:choice>
        <xsd:element ref="block"/>
        <xsd:element ref="data"/>
        <xsd:element ref="field"/>
        <xsd:element ref="var"/>
    </xsd:choice>
</xsd:group>
<xsd:annotation>
    <xsd:documentation>VoiceXML Elements</xsd:documentation>
</xsd:annotation>
<xsd:element name="assign">
    <xsd:complexType>
        <xsd:attribute name="name" type="VariableName.datatype" use="required"/>
        <xsd:attribute name="expr" type="Script.datatype" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="block">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="executable.content"/>
            <xsd:any namespace="##other" processContents="strict"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Form-item.attribs"/>
        <xsd:anyAttribute namespace="##other"  processContents="strict"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="catch">
    <xsd:complexType>
        <xsd:complexContent mixed="true">
            <xsd:extension base="basic.event.handler">
               <xsd:attribute name="event" type="EventNames.datatype"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="choice">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="input"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Cache.attribs"/>
        <xsd:attributeGroup ref="Accept.attrib"/>
        <xsd:attributeGroup ref="Throw.attribs"/>
        <xsd:attribute name="dtmf" type="DTMFSequence.datatype"/>
        <xsd:attribute name="fetchaudio" type="URI.datatype"/>
        <xsd:attributeGroup ref="Next.attribs"/>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="clear">
    <xsd:complexType>
        <xsd:attributeGroup ref="Namelist.attrib"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="else">
    <xsd:complexType/>
</xsd:element>
<xsd:element name="elseif">
    <xsd:complexType>
        <xsd:attributeGroup ref="If.attribs"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="error" type="basic.event.handler"/>
<xsd:element name="exit">
    <xsd:complexType>
        <xsd:attribute name="expr" type="Script.datatype"/>
        <xsd:attributeGroup ref="Namelist.attrib"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="field">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="audio"/>
            <xsd:group ref="event.handler"/>
            <xsd:element ref="filled"/>
            <xsd:element ref="link"/>
            <xsd:element ref="option"/>
            <xsd:element ref="property"/>
            <xsd:group ref="input"/>
            <xsd:element ref="prompt"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Form-item.attribs"/>
        <xsd:attribute name="type" type="xsd:string"/>
        <xsd:attribute name="slot" type="xsd:NMTOKEN"/>
        <xsd:attribute name="modal" type="Boolean.datatype" default="false"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="filled">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="executable.content"/>
            <xsd:group ref="dab.content"/>
        </xsd:choice>
      <xsd:attribute name="mode" type="FilledMode.datatype"/>
        <xsd:attributeGroup ref="RestrictedNamelist.attrib"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="form">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="event.handler"/>
            <xsd:element ref="filled"/>
            <xsd:element ref="initial"/>
            <xsd:element ref="link"/>
            <xsd:element ref="property"/>
            <xsd:element ref="script"/>
            <xsd:element ref="subdialog"/>
            <xsd:group ref="variable"/>
            <xsd:group ref="input"/>
            <xsd:element ref="va:isync"/>
            <xsd:element ref="va:registerlocation"/>
            <xsd:element ref="va:location"/>
        </xsd:choice>
        <xsd:attribute name="id" type="xsd:ID"/>
        <xsd:attributeGroup ref="GrammarScope.attrib"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="goto">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="1">
            <xsd:element ref="va:isync"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Cache.attribs"/>
        <xsd:attributeGroup ref="Next.attribs"/>
        <xsd:attribute name="fetchaudio" type="URI.datatype"/>
        <xsd:attribute name="expritem" type="Script.datatype"/>
```

```
        <xsd:attribute name="nextitem" type="RestrictedVariableName.datatype"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="help" type="basic.event.handler"/>
<xsd:element name="if">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="executable.content"/>
            <xsd:group ref="dab.content"/>
            <xsd:element ref="elseif"/>
            <xsd:element ref="else"/>
        </xsd:choice>
        <xsd:attributeGroup ref="If.attribs"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="initial">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="audio"/>
            <xsd:group ref="event.handler"/>
            <xsd:element ref="link"/>
            <xsd:element ref="property"/>
            <xsd:element ref="prompt"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Form-item.attribs"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="link">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="input"/>
            <xsd:element ref="va:isync"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Cache.attribs"/>
        <xsd:attributeGroup ref="Next.attribs"/>
        <xsd:attributeGroup ref="Throw.attribs"/>
        <xsd:attribute name="fetchaudio" type="URI.datatype"/>
        <xsd:attribute name="dtmf" type="DTMFSequence.datatype"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="log">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
           <xsd:element ref="value"/>
        </xsd:choice>
        <xsd:attribute name="label" type="xsd:string"/>
        <xsd:attribute name="expr" type="Script.datatype"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="menu">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="audio"/>
            <xsd:element ref="choice"/>
            <xsd:group ref="event.handler"/>
            <xsd:element ref="property"/>
            <xsd:element ref="prompt"/>
            <xsd:element ref="va:isync"/>
        </xsd:choice>
        <xsd:attribute name="id" type="xsd:ID"/>
        <xsd:attributeGroup ref="GrammarScope.attrib"/>
        <xsd:attributeGroup ref="Accept.attrib"/>
        <xsd:attribute name="dtmf" type="Boolean.datatype" default="false"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="meta">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:NMTOKEN"/>
        <xsd:attribute name="content" type="xsd:string" use="required"/>
        <xsd:attribute name="http-equiv" type="xsd:NMTOKEN"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="metadata">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:any namespace="##other" processContents="lax"/>
        </xsd:choice>
        <xsd:anyAttribute namespace="##any" processContents="strict"/>
```

```
    </xsd:complexType>
</xsd:element>
<xsd:element name="noinput" type="basic.event.handler"/>
<xsd:element name="nomatch" type="basic.event.handler"/>
<xsd:element name="output" abstract="true"/>
<xsd:element name="option">
    <xsd:complexType mixed="true">
        <xsd:attributeGroup ref="Accept.attrib"/>
        <xsd:attribute name="dtmf" type="DTMFSequence.datatype"/>
        <xsd:attribute name="value" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="prompt" type="speak">
    <xsd:annotation>
        <xsd:documentation>prompt element uses SSML speak type</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="param">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:NMTOKEN" use="required"/>
        <xsd:attribute name="expr" type="Script.datatype"/>
        <xsd:attribute name="value" type="xsd:string"/>
        <xsd:attribute name="valuetype" type="Valuetype.datatype" default="data"/>
        <xsd:attribute name="type" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="property">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:NMTOKEN" use="required"/>
        <xsd:attribute name="value" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="reprompt">
    <xsd:complexType/>
</xsd:element>
<xsd:element name="return">
    <xsd:complexType>
        <xsd:attributeGroup ref="Namelist.attrib"/>
        <xsd:attributeGroup ref="Throw.attribs"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="script">
    <xsd:complexType mixed="true">
        <xsd:attribute name="src" type="URI.datatype"/>
        <xsd:attribute name="srcexpr" type="Script.datatype"/>
        <xsd:attribute name="charset" type="xsd:string"/>
        <xsd:attributeGroup ref="Cache.attribs"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="subdialog">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="audio"/>
            <xsd:group ref="event.handler"/>
            <xsd:element ref="filled"/>
            <xsd:element ref="param"/>
            <xsd:element ref="property"/>
            <xsd:element ref="prompt"/>
        </xsd:choice>
        <xsd:attributeGroup ref="Form-item.attribs"/>
        <xsd:attribute name="src" type="URI.datatype"/>
        <xsd:attribute name="srcexpr" type="Script.datatype"/>
        <xsd:attributeGroup ref="Cache.attribs"/>
        <xsd:attribute name="fetchaudio" type="URI.datatype"/>
        <xsd:attributeGroup ref="Submit.attribs"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="submit">
    <xsd:complexType>
        <xsd:attributeGroup ref="Cache.attribs"/>
        <xsd:attributeGroup ref="Next.attribs"/>
        <xsd:attribute name="fetchaudio" type="URI.datatype"/>
        <xsd:attributeGroup ref="Submit.attribs"/>
    </xsd:complexType>
</xsd:element>
```

```
    <xsd:element name="throw">
        <xsd:complexType>
            <xsd:attributeGroup ref="Throw.attribs"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="data">
        <xsd:complexType>
            <xsd:attributeGroup ref="Cache.attribs"/>
            <xsd:attribute name="name" type="RestrictedVariableName.datatype"/>
            <xsd:attribute name="srcexpr" type="Script.datatype"/>
            <xsd:attribute name="fetchaudio" type="URI.datatype"/>
            <xsd:attribute name="src" type="URI.datatype"/>
            <xsd:attributeGroup ref="Submit.attribs"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="var">
        <xsd:complexType>
            <xsd:attribute name="name" type="RestrictedVariableName.datatype" use="required"/>
            <xsd:attributeGroup ref="Expr.attrib"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="vxml">
        <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
                <xsd:element ref="data"/>
                <xsd:group ref="event.handler"/>
                <xsd:element ref="form"/>
                <xsd:element ref="link"/>
                <xsd:element ref="menu"/>
                <xsd:element ref="meta"/>
                <xsd:element ref="metadata"/>
                <xsd:element ref="property"/>
                <xsd:element ref="script"/>
                <xsd:element ref="var"/>
<xsd:element ref="va:metasync"/>
                <xsd:element ref="va:esync"/>
                <xsd:element ref="va:registerlocation"/>
            </xsd:choice>
            <xsd:attribute name="application" type="URI.datatype"/>
            <xsd:attribute ref="xml:base"/>
            <xsd:attribute ref="xml:lang"/>
            <xsd:attribute name="version" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

# Annex F (informative): Bibliography


- ETSI TS 101 498-2: "Digital Audio Broadcasting (DAB); Broadcast website; Part2: Basic profile specification".
- IETF RFC 1945: "Hypertext Transfer Protocol - HTTP/1.0".
- IETF RFC 2068: "Hypertext Transfer Protocol - HTTP/1.1".
- IETF RFC 1738: "Uniform Resource Locators (URL)".

# History

| Document history | | |
|---|---|---|
| V1.1.1 | November 2008 | Publication |
| | | |
| | | |
| | | |
| | | |