# ETSI TS 101 498-1 V1.1.1 (2000-08)

*Technical Specification*

# Digital Audio Broadcasting (DAB); Broadcast website; Part 1: User application specification

Reference
DTS/JTC-DAB-10-1

Keywords
audio, broadcast, DAB, digital, receiver

***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://www.etsi.org/ipr).

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 [1], 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.

The present document is Part 1 of a multi-part TS covering the DAB Broadcast website, as identified below:

**Part 1: "User application specification";**

Part 2: "Basic profile specification".

# Introduction

The growth of the Internet and the popularity of the "World Wide Web", based on the Hyper Text Transfer Protocol (HTTP) and the Hyper Text Markup Language (HTML), makes web related services an extremely attractive way of providing information to users. The public have already accepted a technology that is now common both in the home and in the workplace, and HTML is a content format that is widely used and supported well by many content creation tools.

The DAB Broadcast website user application gives DAB multiplex operators the opportunity to use HTML as a content format to support information services by using the concept of a "broadcast website". The MOT BWS user application is designed to allow an entire website to be delivered to a receiver using only the broadcast channel of DAB and without the need for any form of return channel.

# 1 Scope

The present document describes the protocol required to create a broadcast carousel of files for a "website". Receivers may then extract information directly from this carousel in order to present the service.

The DAB Broadcast website application applies the DAB-MOT protocol [3] and allows a service provider to deliver HTML content via DAB without the need for a return channel.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

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

[2] ETSI TS 101 756 (V1.1.1): "Digital Audio Broadcasting (DAB); Registered Tables".

[3] ETSI EN 301 234 (V1.2.1): "Digital Audio Broadcasting (DAB); Multimedia Object Transfer (MOT) protocol".

[4] ETSI TS 101 498-2 (V1.1.1): "Digital Audio Broadcasting (DAB); Broadcast website, Part 2: Basic profile specification".

[5] IETF RFC 1945 (1996): "Hypertext Transfer Protocol - HTTP/1.0".

[6] IETF RFC 2068 (1997): "Hypertext Transfer Protocol - HTTP/1.1".

[7] IETF RFC 1738 (1994): "Uniform Resource Locators (URL)".

[8] IETF RFC 2045 to 2049 (1996): "Multipurpose Internet Mail Extensions (MIME)".

[9] IETF RFC 1950 (1996): "ZLIB Compressed Data Format Specification version 3.3".

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

# 3 Abbreviations

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

| | |
|---|---|
| BWS | Broadcast website |
| CA | Conditional Access |
| CGI | Common Gateway Interface |
| DAB | Digital Audio Broadcasting |
| DGI | DAB Gateway Interface |
| FIG | Fast Information Group |
| HTML | Hyper Text Markup Language |
| HTTP | Hyper Text Transfer Protocol |

| | |
|---|---|
| IETF | Internet Engineering Task Force - the IETF is an international body responsible for the development of Internet standards |
| IP | Internet Protocol |
| MIME | Multi-purpose Internet Mail Extensions |
| MOT | Multimedia Object Transfer |
| RFC | Request For Comments - RFCs are used by the IETF for standards and recommendations |
| TCP | Transmission Control Protocol |
| UA | User Application |
| URL | Uniform Resource Locator |
| WIRC | WorldDAB Information and Registration Centre - a WorldDAB office for co-ordinating the technical developments of DAB |

# 4 Syntax specification and maintaining tables of registered values

## 4.1 Syntax specification

The specifications of syntax that appear in the present document are written using a form of pseudo-code that is similar to the procedural language "C"; this provides for easy specification of loops and conditional data structures. Within these specifications, the type of individual data fields is expressed using the mnemonics given in Table 1.

**Table 1: Data type mnemonics for syntax specification**

| Mnemonic | Description |
|---|---|
| uimsbf | Unsigned integer, most significant bit first |
| bslbf | Bit string, left bit first |
| rpchof | Remainder polynomial coefficients, high order first |

## 4.2 Maintaining tables of registered values

The present document contains identifier fields that require values to be registered. Registered value lists associated with data broadcasting specifications for DAB are maintained by the WorldDAB Information and Registration Centre (WIRC). Since the lists and tables contained within the present document might be outdated, please refer to the most recent versions of TS 101 756 [2]. The present document also describes the procedures for registering values in an existing table as well as registering new tables.

# 5 Operation of the MOT BWS user application

## 5.1 The World Wide Web and the Internet

On the Internet, the World Wide Web (WWW) is based largely on the use of HTTP and HTML. HTTP is a protocol that is designed to be used with the TCP/IP protocol stack. It works by using TCP/IP sockets to exchange HTTP requests and HTTP responses and therefore requires a bi-directional communication channel on two levels:

- TCP is a connection oriented protocol that uses a system of acknowledgements and timeouts to implement a reliable, bi-directional stream channel over an IP network. The acknowledgements are used to verify that information has been correctly received and so the protocol inherently requires bi-directional communication, even if the information carried within the stream is uni-directional;

- HTTP is a request-response protocol used for exchanging information in a client-server system. An HTTP client (a web browser) first makes a TCP/IP connection to an HTTP server and requests an object from the server. The server uses the same TCP/IP connection to return the requested data, so HTTP is also necessarily bi-directional in nature.

On the WWW, web browsers make requests for web objects to a web server (HTTP server) which the server usually stores locally in the form of files. This means that an entire website can be represented as a set of files. Because both TCP/IP and HTTP are necessarily bi-directional protocols, it is not possible to use these protocols in a broadcast channel. However, by transporting the complete set of files for the website in the broadcast channel, a web-like experience can be created.

## 5.2 The MOT Broadcast website

The Broadcast website user application uses the MOT protocol to create a broadcast carousel of files for a website. Receivers may then extract information directly from this carousel in order to present the service.

Two distinct classes of receiver are envisaged:

- integrated receivers where native DAB BWS browser software is required to be written for a specific DAB radio platform;
- PC based receivers where the application decoder makes the service available by using the installed desktop web browser.

Because the capabilities of integrated BWS receivers will necessarily be limited by the software installed in them at the time of sale, content intended for such receivers will be required to lie within an appropriate content profile. Services intended for the PC receiver, however, are not limited in this way since the broadcasters choice of content, as on the Internet, is required to be determined by the capabilities of the prevailing browser technology of the time.

The end-to-end architecture of a DAB BWS system is shown in Figure 1.

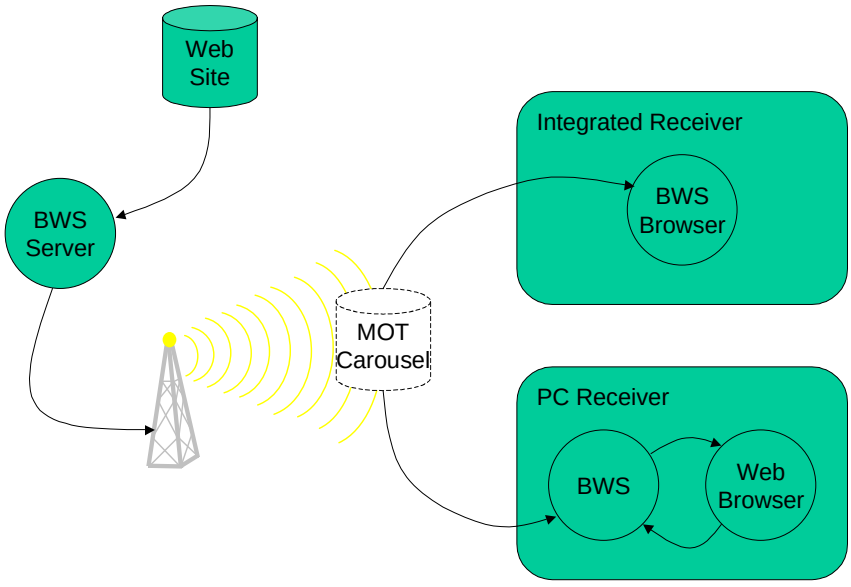


**Figure 1: Operation of the DAB BWS application**

### 5.2.1 The BWS Server

The nature of a website is that the pages contain links to other pages within the site (or possibly to other sites on the Internet or an intranet). Within web pages, these links are encoded as Uniform Resource Locators (URLs).

The BWS server maps the files local to the server that represent the website to be broadcast into an MOT carousel. The files in the carousel are named in such a way that they can be resolved against requests for specific URLs.

### 5.2.2 The application decoder for an integrated receiver

On an integrated receiver, browser software is required to be written specifically for the receiver platform to support the BWS application. Having launched the browser with an appropriate HTML home page, the browser may then follow links embedded in the HTML page by using the URLs of the links to identify other files carried within the MOT carousel.

### 5.2.3 The application decoder for a PC based receiver

On a PC based receiver, the application software for a BWS service does not need to provide all the functionality of a web browser – instead, the software should allow the service to be presented using whatever browser has been installed by the user. This can be achieved by using the set of files for the website, carried in the MOT carousel, to create a real website **local to the receiver**. The software for this performs exactly the same function as a "standard" web server but instead of taking files from the local hard drive, the files are taken from the carousel instead.

In order to gain access to a service in this way, the PC's web browser should request a URL that refers to the local machine, e.g. `http://localhost/` or `http://127.0.0.1/`. It is envisaged that this address could be aliased upon installation of the software to something more appropriate to dab, e.g. `http://www.bws.dab/`.

Note that on a PC receiver, different local TCP/IP ports may be used to provide access to a range of different MOT BWS services, possibly from more than one DAB ensemble.

## 5.3 The BWS MOT carousel

The data for the MOT website shall be carried in a single MOT carousel. Within the MOT carousel, MOT parameters can be associated either with individual objects (by placing them in the MOT Header Extensions), or with the entire carousel (by placing them in the MOT Directory Extension).

The use of the MOT Directory with the BWS user application is **mandatory**.

## 5.4 MOT parameters for individual objects

MOT parameters that are to be applied to individual MOT objects are carried in the MOT header of each directory entry in the MOT Directory.

A summary of the use of MOT parameters for individual objects is given in Table 2, and is specified in detail by the following subclauses. Note that other parameters may be defined within the context of specific profile definition (see clause 7). Any parameters that are encountered that are not understood by a given receiver profile should be ignored. For a description of the currently defined profiles see [4].

**Table 2: Use of MOT parameters for indivdual objects**

| Parameter | Parameter id | Specified in | Mandatory for UA provider | Occurrences |
|---|---|---|---|---|
| RetransmissionDistance | 0x07 | MOT | no | Single |
| Priority | 0x0A | MOT | no | Single |
| ContentName | 0x0C | MOT | yes | Single |
| MimeType | 0x10 | BWS | yes | Single |
| CompressionType | 0x11 | BWS | no | Single |
| AdditionalHeader | 0x20 | BWS | no | Multiple |
| ProfileSubset | 0x21 | BWS | no | Single |
| CAInfo | 0x23 | BWS | no | Single |
| SubscriberInfo | 0x24 | BWS | no | Multiple |

### 5.4.1 The RetransmissionDistance parameter

The use of the `RetransmissionDistance` parameter is optional and its interpretation is defined by the MOT standard.

### 5.4.2 The Priority parameter

The use of the `Priority` parameter is optional and its interpretation is defined by the MOT standard. It may be used to indicate to the MOT carousel decoder the relative cache priority of each object in the carousel, thus it is likely that the home page(s) for the service may be marked with a high priority.

### 5.4.3 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 application decoder shall parse any URL given as part of, for example, an <A> or <IMG> tag within an HTML page and use the resultant path to locate the desired file by matching against the `ContentName` parameter of the files within the MOT carousel. Note that the matching of the character case shall be exact and that the `ContentName` may be preceded by a leading "/".

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.

NOTE: 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 3 shows some example `ContentName` parameters, together with appropriate references.

**Table 3: Example URL references and corresponding `ContentNames`**

| 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` | `%%d.html` | `a/b/%d.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` |

The mapping between the name of the file local to the application server and the value of the name descriptor within the carousel shall be determined by the application server. It is the responsibility of the application server to ensure that any hypertext links within the website are resolved correctly.

### 5.4.4 The MimeType parameter

In HTTP, the type of an object is indicated using the Multi-purpose Internet Mail Extensions (MIME) mechanism. MIME strings categorize object types according to first a general type followed by a specific format, e.g. `text/html`, `image/jpeg` and `application/octet-stream`.

NOTE 1: 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.

In order to correctly present objects from the MOT carousel, it is required to be possible for the receiver to determine the type of the object. Some content types may be signalled using the `ContentType` and `ContentSubType` fields of the MOT headers. However, this mechanism is unsuitable for supporting as yet unspecified MIME types, and so limits the range of types that may be supported *even when the receiver is PC based*. In order to overcome this limitation, the `MimeType` parameter may be used to supply a MIME type string for each object.

NOTE 2: It is not acceptable for the receiver to attempt to determine the type of an object from its "file extension"

Where a `MimeType` parameter is specified for an object within the carousel, the specified type should always be used in preference to the `ContentType` and `ContentSubType` fields of the MOT headers for determining the type of the object. However, where the desired type can be associated with values of `ContentType` and `ContentSubType` from the registered table, service providers should ensure that the correct values are used.

The MIME type associated with the `ContentType` and `ContentSubType` values of 0 and 0 (general_data – Object Transfer) should be assumed to be application/octet-stream.

The MimeType parameter is identified by the ParametId value 0x10 and the parameter data field carries the mime type string appropriate to the object.

### 5.4.5 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. This parameter is identified by the ParamId value 0x11 and the parameter data field carries a single byte identifier (CompressionId) for the compressed data format. If new compressed data formats are to be used, a new CompressionId shall be obtained from and registered with the WorldDAB Information and Registration Centre using the procedures outlined in subclause 4.2.

Table 4 identifies registered compressed data formats correct at the time of publication of the present document, but the list is provided here **for information only** and may be out of date.

**Table 4: Registered compressed data formats**

| CompressionId | Description | Specification reference |
|---|---|---|
| 0x00 | Reserved | |
| 0x01 | gzip compressed data format (Deflate) | RFC1950[9], RFC1952 [10] |

NOTE 1: The gzip file format potentially allows compression algorithms other than the "Deflate" algorithm to be used, although Deflate is currently the only algorithm specified for use with this format.

NOTE 2: A software library to decode gzip files coded with the Deflate algorithm is freely available and may be obtained from:

http://www.cdrom.com/pub/infozip/zlib

### 5.4.6 The AdditionalHeader parameter

In HTTP, a large number of headers may be optionally supplied by a web server in response to a request from a URL. A particularly relevant example of this is the "refresh" header which instructs a web client, such as a browser, to automatically re-request the URL after a specified period of time.

The BWS user application can be used in two receiver environments - the PC decoder and the integrated receiver/decoder. In the PC environment, the AdditionalHeader parameter is used to supply arbitrary additional headers to be included in the HTTP response from the BWS decoder for a given object. For integrated receivers that are dependent on a fixed profile, specific additional headers may be used (e.g. "refresh") but support for any given additional header field will be profile dependent.

The AdditionalHeader parameter is identified by the ParamId value 0x20 and the parameter data field carries an HTTP header field string **without any terminating <LF> or <CR><LF> sequence**. The parameter should be used by either interpreting the header field when the object is requested from the carousel (in an integrated decoder), or by simply incorporating the string into the HTTP header fields written in response to a URL request for the object (in a PC decoder implementation). When including any specified additional header fields in an HTTP response, such header fields should be the last header fields before the blank line that separates the HTTP header fields from the body of the message.

### 5.4.7 The ProfileSubset parameter

Where the carousel for a BWS service carries objects to support more than one BWS profile, additional cache hinting may be applied by the MOT decoder if it knows which profile a given object is used by. For a receiver conforming to profile x, only files that are relevant to profile x receivers need be stored in the cache. The optional ProfileSubset parameter allows the service provider to indicate a list of profiles for which any given object is relevant. The ProfileSubset parameter data field simply carries a list of 1 byte profile ids which identifies *all* of the profiles for which the object is relevant.

If the ProfileSubset parameter is not specified for an object in the carousel, the receiver shall assume that the object may be relevant to all profiles supported by the service.

The ProfileSubset parameter is identified by the ParamId value 0x21.

### 5.4.8 The CAInfo parameter

The `CAInfo` parameter is used to indicate the scrambling status of individual objects within the carousel where a service potentially contains both scrambled and unscrambled objects. The syntax of the `CAInfo` parameter is given in Table 5.

**Table 5: Syntax of the `CAInfo` parameter data field**

| Syntax | Size | Type |
|---|---|---|
| CAInfo_parameter_data_field() { | | |
| **Type of crypto-algorithm** | **6 bits** | **uimsbf** |
| **scrambling_mode** | **2 bits** | **uimsbf** |
| } | | |

**Type of crypto-algorithm**: This field identifies the encryption algorithm used, as defined in EN 300 401 [1], subclauses 9.3.1.1 and 9.3.1.2.

**scrambling_mode**: This field identifies whether the object is "unscrambled", "free access" or "controlled access" according to the values defined in EN 300 401 [1], subclause 9.2.3.

The `CAInfo` parameter is identified by the `ParamId` value 0x23.

### 5.4.9 The SubscriberInfo parameter

If an object within the carousel is scrambled and the receiver is unable to descramble the object, it is desirable for the receiver to be able to present information about how the user may subscribe to the service so that they can descramble any scrambled objects. The `SubscriberInfo` parameter allows this by re-directing the receiver to a replacement object if the receiver is unable to descramble a given object. The syntax of the `SubscriberInfo` parameter is given in Table 6.

**Table 6: Syntax of the SubscriberInfo parameter data field**

| Syntax | Size | Type |
|---|---|---|
| SubscriberInfo_parameter_data_field() { | | |
| **no_ca_flag** | **1 bit** | **bslbf** |
| **no_algorithm_flag** | **1 bit** | **bslbf** |
| **no_subscription_flag** | **1 bit** | **bslbf** |
| **expired_subscription_flag** | **1 bit** | **bslbf** |
| **reserved** | **4 bits** | **bslbf** |
| | | |
| **encryption_specific_flags** | **8 bits** | **bslbf** |
| | | |
| for (i=0;i<N;i++) { | | |
| **content_name_byte** | **8 bits** | **uimsbf** |
| } | | |
| } | | |

**no_ca_flag**: This field indicates if the alternative object identified by the specified `ContentName` should be used if the object cannot be decoded because the receiver does not support conditional access.

**no_algorithm_flag**: This field indicates if the alternative object identified by the specified `ContentName` should be used if the object cannot be decoded because the receiver supports conditional access but does not support the encryption algorithm used for the object.

**no_subscription_flag**: This field indicates if the alternative object identified by the specified `ContentName` should be used if the object cannot be decoded because the receiver supports conditional access but the user has not subscribed to the service.

**expired_subscription_flag**: This field indicates if the alternative object identified by the specified `ContentName` should be used if the object cannot be decoded because the users subscription to the service has expired.

**reserved**: This field should be set to 0.

**encryption_specific_flags**: This field may be used to redirect the receiver to the specified alternative in the event of an error condition defined within the scope of the encryption algorithm used for the object.

**content_name_bytes**: This field identifies the `ContentName` of the object to be used as an alternative to the scrambled object.

Note that the `ContentType` of the replacement object shall be the same as that of the object to which the parameter is associated.

The `SubscriberInfo` parameter is identified by the `ParamId` value 0x24 and may occur more than once for a given object.

## 5.5 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 use of MOT parameters for the entire carousel is given in Table 7, and is specified in detail by the following subclauses. Note that other parameters may be defined within the context of specific profile definition (see clause 7). Any parameters that are encountered that are not understood by a given receiver profile should be ignored.

**Table 7: Use of MOT parameters for the entire carousel**

| Parameter | Parameter id | Specified in | Mandatory for UA provider | Occurrences |
|---|---|---|---|---|
| DirectoryIndex | 0x22 | BWS | yes | Multiple |

### 5.5.1 The DirectoryIndex parameter

The `DirectoryIndex` parameter is used to indicate to the receiver how URLs should be resolved when only a directory is specified. 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 BWS user application, 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.

The syntax of the `DirectoryIndex` parameter data field is given in Table 8.

**Table 8: Syntax of the `DirectoryIndex` parameter data field**

| Syntax | Size | Type |
|---|---|---|
| DirectoryIndex_parameter_data_field() { | | |
| **profile_id** | **8 bits** | **uimsbf** |
| | | |
| for (i=0;i<N;i++) { | | |
| **index_name_byte** | **8 bits** | **uimsbf** |
| } | | |
| } | | |

**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.

The `DirectoryIndex` parameter is identified by the `ParamId` value 0x22 and may occur more than once.

# 6 The BWS decoder

## 6.1 Presenting the service

HTML is a very convenient language for creating a multimedia presentation but it has been developed for use with the HTTP protocol in a network environment. Using HTML as a content format for the MOT BWS user application means that the way the service is presented to the user by an integrated receiver may well be different to the way the user application is decoded and presented on a PC.

### 6.1.1 The PC based BWS decoder

The PC environment comes equipped with standard tools for decoding, presenting and navigating HTML pages - the standard web browser (e.g. Internet Explorer or Netscape). All that is required to decode the service in a PC environment is to make the data from the MOT carousel available to the standard browser in a suitable form.

Web browsers access data on websites using the HTTP protocol which is defined by RFC 1945 [5] (HTTP/1.0) and RFC 2068 [6] (HTTP/1.1). The PC based BWS decoder shall be a web server compliant to at least HTTP/1.0 and shall implement the HTTP methods GET and HEAD. If the DAB Gateway Interface (see subclause 6.4) mechanism is implemented, the POST method should also be supported.

The PC based BWS decoder is not required to implement the keepalive function of HTTP (which permits multiple resource requests and responses to be sent using a single TCP/IP connection) but it should be noted that better performance can be achieved using this feature.

The BWS decoder shall respond to requests for URLs by attempting to locate a requested resource (file object) within the MOT carousel. The way in which a URL is resolved against objects carried within the MOT carousel is defined in subclause 6.2.

NOTE: The PC decoder is not required to check the scheme and host_name components of the requested URL.

### 6.1.2 The integrated BWS decoder

On an integrated receiver with a native platform, it is not possible to use standard browser software and so both the HTML decoding and presentation functions and the MOT carousel decoding functions shall be implemented specifically for the receiver platform.

If the receiver software is written to run on top of an embedded Operating System (OS) with support for TCP/IP, it may well be attractive to structure the software in exactly the same way as it is structured on the PC. Although this might cause the software to have a slightly larger footprint within the receiver, the increased modularity of the code will almost certainly be easier to write and maintain, as well as allowing standard code libraries to be used for the various modules.

If no OS is used, or there is no support for TCP/IP, it is likely that the functions of browser and MOT carousel interface will be combined into a single module. This means that, instead of parsing a URL within an HTML page and issuing an HTTP request to a web server, the BWS browser shall make a direct request to the carousel for the resource identified by the URL.

## 6.2 Resolving URLs

The common format of a complete URL is:

`scheme:scheme_specific_part`

NOTE: The common internet scheme syntax is:

scheme://user:password@host:port/path;parameters?query#fragment.

For HTTP, this becomes:

`http://host_name/absolute_path`.

Where a hyperlink is required from one page within a BWS service to another page within the service, the only supported scheme is HTTP and the host_name component of the URL is not required to be specified. Thus all links to content within the service should be made using a relative URL (RFC 1738 [7]), since the service cannot be legitimately identified as being hosted by any specific Internet address. In practice, this means that such links will be either relative or absolute paths only.

For a PC based decoder, the parsing of URLs within an HTML page is handled by the browser, and so is outside the scope of the BWS user application software. This means that if an Internet connection is available, a Web browser will automatically follow links from a BWS service that point to Internet resources. Because the parsing of URLs is performed by the web browser rather than the BWS decoder, there is no impact on the software for decoding the BWS service when services contain such links. On an integrated receiver it should be recognized that it is possible that URLs that refer to Internet resources may be encountered even though they are not supported by the signalled profile. In such a case, they should be handled in the same way as any other invalid URL whose target cannot be located within the MOT carousel.

For profiles that do not permit references to network resources, any URL that specifies a scheme other than `http` shall be considered invalid. In the same way, any URL that specifies a host shall also be considered invalid.

### 6.2.1 Handling invalid URLs

If any invalid URL is encountered, the decoder shall behave in one of two ways:

- if the URL is in an embedded object tag, such as an inline image, the receiver shall behave as for an object that cannot be found within the carousel;
- if the URL is a Hypertext link, the link text shall be treated as ordinary text rather than as a link.

### 6.2.2 Handling absolute and relative paths

When URLs are used to request objects from the MOT carousel, the data for the requested object should normally be identified by a relative URL containing just an absolute or relative path. The decoder should first parse the URL according to RFC 1738 [7] and then match the resulting path against the list of `ContentName` parameters for the carousel (the leading "/" may be omitted from the `ContentName` parameter). In certain circumstances, however, this mechanism is not sufficient - there are two specific exceptions:

- URLs that identify a sub-directory within the service (i.e. "`/xyz/`" or "`/xyz`", where `xyz` refers to a directory). Note that this might be the root of the service, i.e. "/";
- URLs that are reserved for DAB Gateway Interface functions (see subclause 6.4).

### 6.2.3 Handling requests for URLs that refer to directories

If a request is made for an object with a URL that only specifies a directory within the service (including URLs referring to the root directory, "/"), the object to be returned from the carousel shall be determined by appending the name of index file specified by the `DirectoryIndex` parameter with the appropriate profile_id.

For example, if two profiles of the service are supported, two `DirectoryIndex` parameters may be specified:

- Profile 1 - index1.html
- Profile 2 - index2.html

Given a request for the relative URL `/pqr/xyz/`:

- the profile 1 receiver will attempt to find the object with the `ContentName '/pqr/xyz/index1.html'` within the MOT carousel;
- the profile 2 receiver will attempt to find the object with the `ContentName '/pqr/xyz/index2.html'` within the MOT carousel.

NOTE: Receivers should be aware that services may include URLs of the form '/pqr/xyz' (i.e. without the trailing "/") where "xyz" corresponds to a directory. Because of this, receivers should always make sure that such references are not directory references before declaring them invalid.

It shall be mandatory for a `DirectoryIndex` parameter to be provided corresponding to the lowest profile signalled for the service in FIG 0/13 (see EN 300 401 [1]).

### 6.2.4 Starting the service

The service shall always be started by using the relative URL "/" in accordance with subclause 6.2.3. This means that the `DirectoryIndex` parameter with the most appropriate profile_id shall be used to determine which object from the MOT carousel the receiver should use to start the service.

If no `DirectoryIndex` parameter is specified that can be used for the profile of the application supported by the receiver, the receiver shall behave as for any other object that cannot be found within the carousel.

### 6.2.5 Error handling, including requests for objects that do not exist

Due to the different models of operation between a PC based decoder implementation and a receiver native implementation, different approaches are required to handling errors when requesting objects from the carousel. The most obvious problem that may be encountered is a request for an object that can't be found within the carousel. Service providers should, of course, try to ensure that this does not happen, however steps should be taken to handle this event if it does occur.

On a PC where the decoder provides access to the service through HTTP, HTTP itself provides an error reporting mechanism. In the case of a URL request that cannot be satisfied, web servers should respond with status code of 404 (Not Found), together with an HTML page explaining he problem. MOT BWS decoders implemented on a PC should be fully compliant with, at least, the specification for HTTP1.0, and should report all errors using the appropriate status code. The only other HTTP status code applicable to problems with the MOT carousel is 503 (Service Unavailable), which should be returned when the whole carousel is unavailable for any reason (e.g. a BWS service has not been selected).

On an integrated receiver with native browser software, a failure to resolve a given URL should be presented in a way determined by whether the request is for a new page or merely for an object embedded within a page. If the request was for a new page (resulting from following a link, for example), a message should be displayed indicating that the requested page cannot be found. Note that there is no requirement to indicate to the user a status code of 404 (although this is not forbidden). For an inline image embedded within a page, however, a suitable icon should be displayed in place of the object indicating that that part of the page cannot be rendered.

## 6.3 Determining object type

### 6.3.1 Using the MimeType parameter

Because the use of the `MimeType` parameter does not constrain the range of types that may be signalled, the `MimeType` parameter is the preferred mechanism for signalling object type and the `MimeType` parameter is mandatory for all objects in the carousel. The meaning of the MIME type strings should be determined according to the list of types registered by the IETF as defined in RFC 2045-49 [8].

The use of the `MimeType` parameter for all objects within the MOT carousel is mandatory. Where possible, however, the `ContentType` and `ContentSubType` fields shall be set to appropriate values that correspond to the given MIME type string.

### 6.3.2 Using the ContentType and ContentSubType fields

For reasons of backwards compatibility, the `ContentType` and `ContentSubType` fields shall always be given values appropriate to the MIME type string given in the `MimeType` parameter. If there are no values for the `ContentType` and `ContentSubType` fields that match the desired MIME string, both fields shall be set to 0 to indicate general data.

## 6.4 The DAB Gateway Interface

Web servers on the Internet generally respond to requests for URLs by returning the file data corresponding to the request. However, when servers need to provide dynamic data – such as database queries or "web-cam" pictures – the Common Gateway Interface (CGI) system is used. CGI allows web servers to execute sever-side scripts to generate content rather than simply returning a static file, and also allows parameters to be passed to the script so that HTML can be used as a user interface to client-server systems. The most common example of the use of CGI are search engines such as Yahoo or Infoseek, where the user enters a search string into an HTML form; the search string is processed by a CGI script which then returns an HTML page containing the results of the search.

There are two important elements to the CGI mechanism:

- identifying that a URL corresponds to an executable program;
- passing parameters to the CGI script through a query string.

CGI scripts are usually identified by web servers using one of two methods: either through the directory name identified by the path of the request URL or by the file extension of the requested URL. The most common examples are URLs of the following forms:

- `http://www.website.domain/cgi-bin/xxx`;
- `http://www.website.domain/some-path/xxx.cgi`.

Parameters are then passed to these scripts through a query string which is appended to the request URL separated by a "?". The forms feature of HTML is designed to automatically construct search strings of the form `?parameter=value&parameter=value&`… so that these can then be submitted for form processing by a CGI script.

In the context of the MOT BWS application it is impossible for service providers to write programs to run a wide variety of receiver platforms – since the web server is conceptually located on the receiver, that is where a CGI script would have to be run. However, if certain CGI programs are pre-defined and could be assumed to be implemented by the web server it would then be possible to write HTML using those programs. The service provider would not need to write the programs themselves as they would be written alongside the server itself, by the receiver manufacturer.

There are a number of DAB specific functions that one might want to perform:

- receiver tuning;
- service selection;
- volume control.

Inevitably, the key to being able to implement such functionality is defining the calling interface – which, in the case of CGI, is the query string. If the BWS were to define CGI like functions to support, for example, receiver tuning, it would then be possible for the author of the receiver software to implement these functions as long as the format of the query string were defined.

In order to support such a facility within the context of the MOT BWS, what is required is a mechanism for identifying a particular URL within the BWS as being a CGI like function. This is done by reserving the "directory" `/dgi-bin/` for "DAB Gateway Interface" (DGI) functions. Interfaces to certain receiver functions are then defined by:

- specifying the name of the DGI function;
- specifying the format of the associated query string.

For example, one might consider defining a service selection function as:

`/dgi-bin/select_service?service_id="12345"`.

Such a reference appearing as a link in an HTML page of the BWS service could then be used to automatically re-tune the receiver when the link is activated.

In order for such a mechanism to be viable, it is essential that the paths for any DGI functions that may be defined should not overlap with the paths of files broadcast in the MOT carousel. Thus ContentNames of the form `/dgi-bin/`… are **prohibited**.

No DGI functions are defined in the present document. Any DGI functions shall be specified separately as part of a profile specification (see subclause 7.1).

# 7 Application signalling

## 7.0 General

The use of the Broadcast website user application within a DAB data channel shall be indicated by the use of FIG0/13 (see EN 300 401 [1]) with a `UserApplicationType` value of 0x002. The user application data field shall carry a one byte field - the `MinimumProfileId` - indicating the minimum profile of BWS decoder that should attempt to decode the service. Receivers should ignore any user application data following the `MinimumProfileId` field.

The profile identified the `MinimumProfileId` field shall be determined from the registered table maintained by WIRC.

Table 9 identifies registered profiles correct at the time of publication of the present document, but the list is provided here **for information only** and may be out of date. For a description of the defined profiles see [4].

**Table 9: Registered BWS profiles**

| Profiled | Description | Specification reference |
|---|---|---|
| 0x00 | Reserved | |
| 0x01 | Basic Integrated Receiver Profile | see 7.2.1 |
| 0xFF | Unrestricted (PC) Profile | see 7.2.2 |

## 7.1 Specifying BWS user application content profiles

In order to guarantee that a broadcast service conforming to a particular profile will always be decodable by a receiver that implements BWS software for this profile, it is essential for the profile to be a complete specification for all parameters that affect the presentation of the service.

In general, this will require the specification of both Quality Of Service (QOS) parameters as well as Fact Of Service (FOS) parameters.

As future profiles of the BWS user application may introduce new features that shall be constrained, it is not possible to give a definitive list of all parameters that shall be defined. However, the parameters given in the following subclauses shall be described, even if there is no constraint applied (applicable to profiles intended for a generic PC based decoder).

### 7.1.1 Fact of service parameters

Fact of service profile parameters are parameters that determine whether or not it is possible to decode and present the service. Any feature of the BWS user application that shall be constrained in order for a receiver to be guaranteed to be able to decode the service shall be included in a profile specification. As the features of the BWS user application expand, new "fact of service" profile parameters may need to be considered for new profile specifications (e.g. as a result of the definition of new MOT parameters). The list of profile parameter types given below is intended as guide but is not necessarily an exhaustive list.

#### 7.1.1.1 Supported content types

As integrated receivers will necessarily support a finite set of content formats, the range of supported content types shall be completely specified within a profile specification. For example, it is likely that the content type `text/html` will be supported, together with a variety of image types.

Not that for a given content type, it may also be necessary to define an unambiguous profile of the content type - particularly the `text/html` type.

#### 7.1.1.2 Supported profile of HTML

Profile specifications should indicate the precise HTML syntax and semantics that can be successfully parsed and rendered by decoders. It should be noted that the general philosophy behind HTML is that unrecognized HTML tags should not cause the parsing of an HTML page to fail, if at all possible.

When detailing the HTML tags that are supported by a given profile, care should be taken to ensure that the behaviour of the receiver in response to all tag parameters is completely and unambiguously defined.

#### 7.1.1.3 Supported additional HTTP header fields

Where a profile allows use of the `AdditionalHeader` parameter, integrated receivers are required to know which additional headers are supported by a given profile. If unrecognized additional header fields are specified for a service that is signalled to conform to a profile that does not require support for them, then the additional header fields should be ignored by the receiver.

#### 7.1.1.4 Maximum object size

It is likely that integrated receivers will have limited memory available and that there will be a limit on the size of the largest object in the carousel that can be successfully decoded by the receiver. In profiles for receivers that are likely to be short of memory, the maximum allowed size of any object in the carousel *for that profile* should be specified.

#### 7.1.1.5 Maximum total size of all objects rendered within a page

As with the maximum object size, receivers with limited memory may have difficulties presenting an HTML page with a series of objects whose individual sizes lie within the maximum object size, but whose total size is considerably larger. Profile specifications may choose to include a specification for the maximum total size of all objects that are referenced when rendering an HTML page in order to avoid problems.

### 7.1.2 Quality of service parameters

Quality of service parameters are relevant when the content for a service can be successfully interpreted by the receiver software without error, but where the resulting presentation is unacceptably degraded if the receiver cannot meet the requirements of the profile parameter.

As with the fact of service parameters, it is not possible to give a definitive list here but the following subclauses describe some of the quality of service parameters that should be considered when defining profiles.

#### 7.1.2.1 Minimum receiver cache size

The MOT carousel can be used to deliver a set of files in a broadcast Digital Radio channel and can operate entirely without cache memory, if desired. However, the effect of cache memory is to improve the performance of the carousel with respect to carousel access time, and so, affects the perceived quality of the service. Note that cache memory cannot improve service acquisition time.

In order to guarantee a certain level of service performance, the minimum amount of receiver cache memory (possibly zero) should be specified as part of a profile definition.

#### 7.1.2.2 Minimum receiver display characteristics

Whilst it may be possible for an HTML page to be rendered into a display buffer of some form, the characteristics of the display device will determine whether or not the resulting presentation quality is acceptable for the service. For example, a colour image may be able to be rendered satisfactorily on a monochrome display but may end up displayed as a solid black image on a black and white display. Similarly, a display with insufficient resolution may provide an unacceptable presentation.

Where minimum display characteristics are specified, *at least* the following parameters should be considered:

- display size in pixels;
- colour depth.

Given the nature of HTML content, it may also be desirable to impose a limit on the amount of content that can be rendered "off-screen" and accessed through some form of scrolling mechanism. This allows receiver manufacturers to provide an appropriate "feel" to the user interface.

#### 7.1.2.3 Maximum carousel period

It may be desirable to ensure that for a given profile, the user expectation of a service's performance is independent of the particular service provider. Thus, for certain profiles, it may be desirable to specify a maximum carousel period to avoid large differences in acquisition time between services and between service providers.

## 7.2 Application profile specifications

The present document assumes the definition of at least two distinct profiles - the Basic Integrated Receiver profile and the Unrestricted (PC) profile. These two profiles are described briefly below but are the subject of separate specification documents.

### 7.2.1 Basic Integrated Receiver content profile

The "Basic Integrated Receiver" profile specification is aimed at the first BWS receivers to be launched into the market. Such receivers will be based on a "quarter VGA" display format and will have limited processing power and memory capacity. The complete profile specification is defined in Part 2 of the MOT Broadcast website specification (Basic Profile Specification) [4].

### 7.2.2 Unrestricted (PC) content profile

When delivering services to a PC based decoder, the part of the receiver responsible for presenting he service (i.e. the web browser) is developed and upgraded independently of the BWS user application. Thus, it is not appropriate to constrain the nature of the content for services intended for PC based decoders.

# 8 Restrictions that apply to pilot project receivers

#### MOT headers

It should be noted that pilot project receivers do not decode the mandatory MOT Directory and so it is essential to provide MOT headers when targeting services at such receivers.

#### The Label parameter

The `Label` parameter (`ParameterId` = 0x0B) is used to identify the initial object to be used to boot the application **by original MOT receivers**. The use of the label parameter is superseded by the `DirectoryIndex` parameter, carried in the MOT Directory Extension.

The `Label` parameter is mandatory when the service is required to support pilot project receivers.

#### The VersionNumber parameter

The `VersionNumber` parameter is mandatory when updating content for services targeted at pilot project receivers.

#### The ContentName parameter and URL resolution

When supporting services to pilot project receivers, care must be taken to ensure that **no** absolute paths are specified in HTML hyperlinks. This is because there is a danger of them being referenced to the root directory of a local filing system rather than the root of the carousel. It should also be noted that pilot project receivers will not behave correctly if references are made to a directory since they will not decode the `DirectoryIndex` parameter.

Pilot project receivers require `ContentNames` with a leading "/".

# History

| Document history | | |
|---|---|---|
| V1.1.1 | August 2000 | Publication |
| | | |
| | | |
| | | |
| | | |