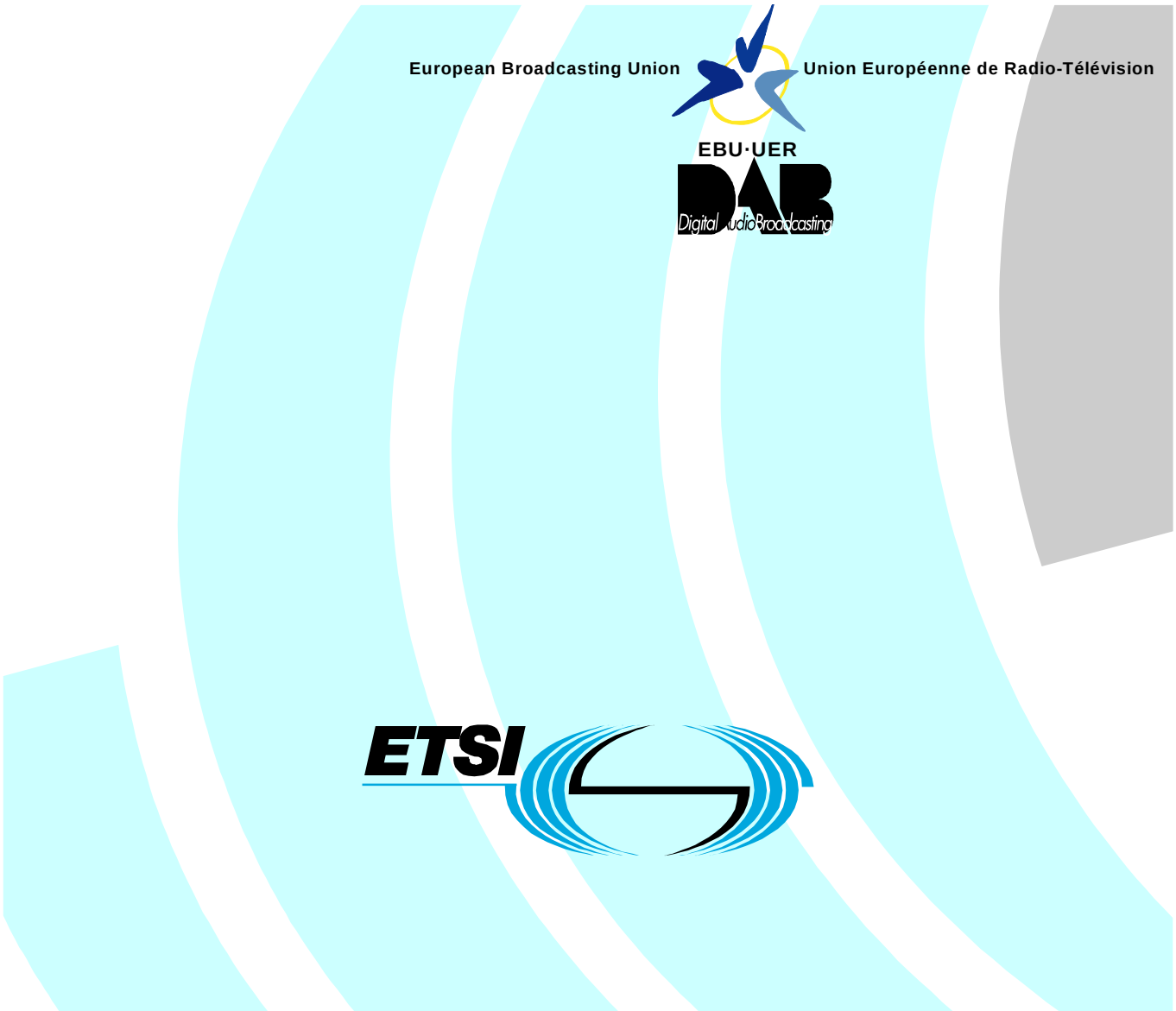
ETSI TS 101 498-1 V2.1.1 (2006-01)

Technical Specification

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



Reference
RTS/JTC-DAB-43

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

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

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 [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 deliverable covering Digital Audio Broadcasting (DAB); Broadcast website, as identified below:

Part 1: "User application specification";

Part 2: "Basic profile specification";

Part 3: "TopNews 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.

Version 1.1.1 of the present document lists restrictions that apply to pilot project receivers. Since these old receivers are no longer relevant, the corresponding clauses are no longer part of this revision of the present document. The readers should refer to the earlier version (V1.1.1) of the present document for more details regarding the support of these outdated receivers. The MOT parameter `Label` is also no longer mentioned in the present version of the BWS standard; see the earlier version (V1.1.1) of the present document and V1.2.1 of the MOT standard for a description of this MOT parameter and its use by the old pilot project receivers.

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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NOTE: 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

4 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

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. Also PDA systems and high-end mobile phones will probably already provide a web browser. In this case the software architecture for the BWS receiver will most probably follow the same rules as the PC based receivers.

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.

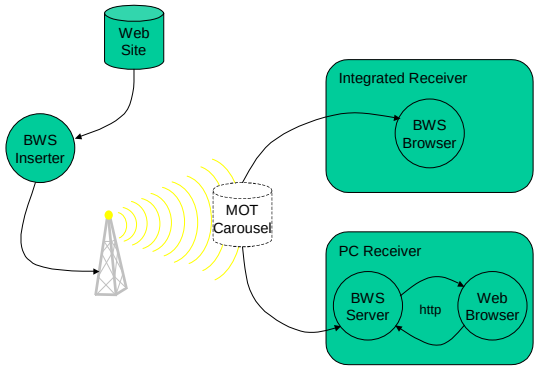


Figure 1: Operation of the DAB BWS application

5.2.1 The BWS Server for the PC receiver class

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 is the interface between the MOT decoder and a standard web browser. The BWS Server maps requests of the Web Browser for specific URLs to objects within the MOT carousel. To assure that the full functionality of the Broadcast website can be provided, it is mandatory for the BWS server to have an HTTP interface, see RFC 1945 [5] and RFC 2068 [6].

Whenever an object is requested by the Web Browser, for instance if a link within the currently presented web page is selected, then the Web 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 BWS Server.

The BWS Server will map the requested absolute path to an object within the MOT carousel with the corresponding `ContentName`. The object will then be returned the Web 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 Web Browser. This information is important for cache management;
- if an invalid or yet unavailable object is requested by the Web Browser, then an appropriate error message can be returned to the Web Browser. It is also possible to tell the Web Browser to retry loading an object that is not yet available;
- by using the appropriate http parameters, the Web Browser can be told to automatically recheck for updates of the currently displayed web page at regular intervals.

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 entry page of the service (see clause 7), 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 Broadcast website user application uses MOT directory mode. The BWS profile determines if MOT directory compression is permitted.

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 information of an object), or with the entire carousel (by placing them in the MOT directory extension).

The BWS provider shall ensure that the links from one object of the MOT data carousel to another object of the MOT data carousel shall be partial (e.g. "../images.jpg" or "/index_pc.html"). 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).

NOTE: The BWS provider assures that no links to non-existent files are contained within the files of the data carousel. However, during updates of the data carousel it is still possible that the browser requests an object that is no longer available. The BWS receiver must be able to cope with this eventuality (e.g. by redirecting the user to the entry page of the BWS).

5.4 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.

A summary of the use of MOT parameters for individual objects is given in table 2, and is specified in detail by the following clauses. Implementation tips are given in annex A. For details regarding all parameters specified in the MOT standard refer to [3].

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 shall be ignored.

Table 2: Use of MOT parameters for individual objects

Parameter	Parameter Id	Specified in	Mandatory for UA Provider	Mandatory for Receiver	Occurrence
ContentName	0x0C	MOT [3], present document	Yes	Yes	Single
MimeType	0x10	MOT [3]	Yes	Yes	Single
CompressionType	0x11	MOT [3]	No	Yes	Single
AdditionalHeader	0x20	present document	No	Yes	Multiple
ProfileSubset	0x21	MOT [3]	No	No	Single

The MOT functionality "caching support" is optional (but recommended) for both BWS provider and receiver.

Collecting MOT body segments whose TransportId is not described in the MOT directory (see EN 301 234 [3]) will significantly improve receiver start-up and permit faster changes to the MOT data carousel. Therefore it is strongly recommended for all receivers.

The MOT functionality "conditional access on MOT level" is optional for both BWS provider and receiver. However, every MOT decoder shall check if the MOT parameter `CAInfo` is signalled for an MOT object, thus indicating that the object is scrambled. A receiver with an 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 parameter `CompressionType`. Every MOT decoder shall be able to detect and discard MOT bodies it cannot uncompress. The BWS profile determines if and which MOT compression schemes the MOT decoder of the receiver shall support.

5.4.1 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 BWS decoder shall parse any URL given as part of, for example, an <A> or <IMG> tag within an HTML page 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.

NOTE 1: This restriction is used to simplify handling of URLs that refer to directories, see clause 6.2.3.

`ContentName`s shall not start with "dgi-bin/", see clause 6.4 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.

NOTE 2: 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 examples `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`	`%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`

5.4.2 The MimeType parameter

The `MimeType` parameter indicates the type of an object using the Multi-purpose Internet Mail Extensions (MIME) mechanism (see RFC 2045 [8] to 2049 [12]). 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: 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. Refer to [3] for details on how to set the MOT header parameters `ContentType` and `ContentSubType`.

5.4.3 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. The BWS profile determines if and which MOT compression schemes the MOT decoder / user application decoder shall support.

5.4.4 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. A particularly relevant example of this is the "Refresh" parameter in the http response header that instructs a web client, such as a browser, to automatically re-request the URL after a specified period of time.

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

5.4.5 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.

5.4.6 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.

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 4, and is specified in detail by the following clauses. Implementation tips are given in annex A. 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 shall be ignored.

Table 4: Use of MOT parameters for the entire carousel

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

The MOT functionality "caching support" is optional (but recommended) for both BWS provider and receiver.

Sorting of MOT header information (signalled by the MOT parameter `SortedHeaderInformation`) is recommend for the BWS provider.

5.5.1 The DirectoryIndex parameter

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 BWS, 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.

NOTE: An empty path ("/") identifies the initial object of a BWS. 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 BWS profiles might have different initial objects) must be in the root directory of the BWS.

The syntax of the `DirectoryIndex` parameter data field is given in table 5.

Table 5: 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. 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 EN 300 401 [1]).

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 means that the way the service is presented to the user by an integrated receiver may well be different to the way the BWS 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 clause 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 clause 6.2.

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, partial links shall be used. Thus all links to content within the service should be made using a partial URLs (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 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. However, it is recommended that user be told that the erroneous link points outside the BWS data (e.g. by telling the user that links of a certain type ("http", "ftp", "mailto", etc.)) are not supported or that links to the Internet require the establishing of an Internet connection first).

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 an invalid URL is requested by the web browser, the browser shall react in the following way:

- if the URL is in an embedded object tag, such as an inline image, the decoder shall behave as for an object that cannot be found within the carousel (e.g. display a "broken image" (a special replacement image that is used to indicate that the requested image is not available));
- if the URL is a Hypertext link that points outside the BWS (e.g. to the Internet), and the referenced resource can not retrieved by the web browser, then an appropriate error message shall be presented informing the user that the referenced resource is unavailable; see also clause 6.2.

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 partial 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, see annex A. 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. "/", see annex A.
- URLs that are reserved for DAB Gateway Interface functions (see clause 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.

6.2.4 Starting the service

The service shall always be started by using the relative URL "/" in accordance with clause 6.2.3. This means that the `DirectoryIndex` parameter with the appropriate ProfileId 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 any of the BWS profile(s) supported by the receiver, then the receiver shall behave as for any other object that cannot be found within the carousel.

NOTE: This should never happen since the content provider must provide a `DirectoryIndex` parameter for all BWS profiles signalled in FIG 0/13.

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 not 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 200 (OK), together with an HTML page explaining he problem.

NOTE: A status code of 404 (Not found) might sometimes seem more appropriate, but some browsers will then not display the provided error HTML page, but use a browser specific error page.

All error pages should contain a link that points to an existing page of the BWS data, for instance to the entry page. It is recommended that all warning pages that indicate that a requested object is not yet available (see annex A) be automatically replaced by the requested object as soon as it becomes available.

MOT BWS decoders implemented on a PC should be fully compliant with, at least, the specification for HTTP1.0. The only HTTP status code applicable to problems with the MOT carousel is 503 (Service Unavailable), which could 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 ("broken image") should be displayed in place of the object indicating that that part of the page cannot be rendered.

6.3 Determining object type

Because the use of the `MimeType` parameter does not constrain the range of types that may be signalled, the `MimeType` parameter is the 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 [8] to RFC 2049 [12].

The use of the `MimeType` parameter for all objects within the MOT carousel is mandatory.

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 server-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 Google, 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/script;
- 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 on 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. Service providers 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 (i.e. by reserving `ContentNames` that start with "dgi-bin/"). 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 `ContentName`s shall not start with "`dgi-bin/`".

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

NOTE 1: This interface can be used to support DAB URLs on PC based terminals. To do this, the BWS server has to rewrite all HTML pages it returns to the web browser so that all DAB URLs within the new (rewritten) HTML pages now start with a reference to an appropriate CGI interface on the BWS server. For instance all DAB URLs in the received MOT objects could be rewritten from "<DABURL>" to "/dgi-bin/x-daburl?url=<DABURL>".
If the user selects such a rewritten link, then the browser will contact the DAB gateway interface and provide the selected DAB URL as a parameter to the BWS server. The BWS server can then react accordingly, for instance by tuning to another ensemble and then sending an HTTP response to the web browser redirecting it to another HTTP server (e.g. another BWS server).
Note that it is the BWS profile that defines the types of DAB URLs the receiver has to support.

NOTE 2: A BWS receiver that does not use HTTP to access the MOT objects (this could for instance also be a PC that uses a special plug-in for the browser) will usually directly support DAB URLs and such a receiver will thus not need to rewrite DAB URLs inside the received HTML objects to support the DAB URLs.

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] and TS 101 756 [2]) with a `UserApplicationType` "MOT Broadcast Web Site".

The user application data field for the Broadcast website user application is a sequence of 1-byte values, each being a `ProfileId`, indicating which profile(s) of the Broadcast website user application are carried there. If there is more than one `ProfileId`, then the list shall be sorted in ascending order with the lowest `ProfileId` first.

If other BWS specific parameters than the list of profiles is carried in the user application data field, then there shall be a single 0x00 after the end of the list of profiles and before the first other BWS specific parameter.

NOTE: The decoder can easily extract the list of profiles from the user application data field. The list of profiles either ends at the end of the user application data field (no other BWS specific parameters) or at a delimiting 0x00 (other BWS specific parameters follow after the 0x00), whichever comes first.

The `ProfileId`s are defined in table 6. Any remaining values for the `ProfileId` are reserved for future use and shall be discarded by receivers:

Table 6: Registered BWS profiles

ProfileId	Description	Specification reference
0x00	Reserved	
0x01	Basic Integrated Receiver Profile	See clause 7.2.1
0x02	TopNews Profile	See TS 101 498-3 [13]
0xFF	Unrestricted (PC) Profile	See clause 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 clauses 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.

Note 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 clauses 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.

Compression on MOT object level (MOT parameter `CompressionType`) is permitted and the BWS receiver shall at least support "gzip" decompression with all gzip window sizes up to 32 Kbytes.

Compression of the MOT directory is not used (i.e. an uncompressed MOT directory must be broadcast). However, in addition a compressed MOT directory may be broadcast.

Annex A (informative): Implementation tips

The BWS user application can be used in two receiver environments - the PC decoder and the integrated receiver/decoder.

This annex gives some tips how MOT parameters can be processed on these two types of receivers. However, it will still be necessary to check RFC 2068 [6] and EN 301 234 [3] for details.

A.1 General tips

A.1.1 Mapping of URLs to ContentNames

The following list explains the necessary steps to map a URL path to an MOT object of the data carousel.

NOTE 1: The web browser is responsible for URL unquoting (for instance "/we%20are%20%2f1" would be unquoted to "/we are #1", and the latter would be requested); therefore the mapping steps describe below do *not* include any unquoting.

NOTE 2: The browser is also responsible for translating partial URLs to absolute URLs. All requested URLs will thus be absolute and also always start with a leading "/". However, the `ContentNames` used by the BWS do not use leading slashes. Therefore the leading slash of the request has to be ignored when looking up the MOT object with the appropriate `ContentName`.

The requested URL path could end with a trailing "/". In this case the mapping shall assume that the URL path refers to a directory, see clause 6.2.3. So processing will directly start with step 2.

1) The URL path most probably directly references the `ContentName` of an object.
   If the mapping does not provide an MOT object with the appropriate `ContentName`, then it has to be checked if the URL refers to a directory (step 2).

2) It is checked, whether the URL path references a directory. The mapping takes the requested URL path; and appends a delimiting "/" (unless the requested URL path already ends with a "/"). It then takes the content of the MOT directory extension parameter `DirectoryIndex` for the current receiver profile and appends it as well.
   The mapping again has to look up the MOT object with the appropriate `ContentName`s and return the object (if available).

3) If no object can be returned (the object can not be found in the current MOT directory, or it is impossible to decompress or descramble the object), then an appropriate error message shall be returned. It is recommended telling the user that a page was requested that is no longer (or never was) available. It is recommended that a substitute page is returned containing a reference to a valid web page, for instance to the entry page of the BWS.

4) If the requested object could be found (i.e. it is part of the current MOT directory), but is not yet available (e.g. not yet reassembled or not yet descrambled), then a warning page should be returned that explains that the object exists but is not yet available. The user could then just wait or select another link. When the requested object becomes available, it should automatically replace the warning page.

Table A.1: Example mapping from requested URL path to ContentNames

Requested URL path	DirectoryIndex value for receiver profile	ContentNames that have to be looked up by the mapping	Remarks
/	index_pc_profile.html	index_pc_profile.html	Mapping starts with step 2
/logo.jpg		logo.jpg logo.jpg/index_pc_profile.html	
/news		news news/index_pc_profile.html	
/news/		news/index_pc_profile.html	Mapping starts with step 2

A.2 Integrated BWS receiver

The assumption in this clause is that the integrated receiver combines web browser and MOT decoder. Therefore there will be no standard interface (such as http) between the browser and the MOT decoder. Communication between both components could for instance be made using function calls or inter process communication (events, shared memory, etc.).

However, some integrated receivers might use http as the protocol between the browser and the DAB specific modules. In this case the implementation tips for PC based receivers apply.

A.2.1 Mapping URLs to ContentNames

If the user selects a link, then the integrated receiver must first convert the link to its absolute form (an absolute URL). Usually this is the task of the browser. It is recommended that the resulting absolute URL has a leading "/" (as it is used in http). The absolute link is then mapped to an MOT object (see clause A.1.1).

The error page (object not inside the current MOT directory) should tell the browser to automatically reload the entry page of the BWS after some seconds; the page could also contain a link that references the entry page of the BWS.

The warning page (object not yet available) should also tell the browser to automatically try reloading the page requested by the user after some seconds. The same warning page will be returned for every reload request until the requested page is finally available or the page is removed from the MOT directory. In the latter case an error page (see above) will be returned.

A.2.2 The MOT parameter MimeType

The integrated receiver must parse this parameter and inform the web browser about the type of the requested object. The integrated receiver must take into account that the MimeType parameter may contain additional parameters (such as the character set that is used inside the MOT object).

A.2.3 The MOT parameter CompressionType

All compressed objects must be decompressed before presentation. Decompression could be done by the MOT decoder or by the browser.

A.2.4 The MOT parameter AdditionalHeader

For integrated receivers that are dependent on a fixed profile, specific additional headers may be used, but support for any given additional header field will be profile dependent.

The parameter data field will be interpreted when the object is requested from the carousel.

NOTE: The integrated receiver uses a special browser to display the objects. The MOT decoder can directly inform such a special browser when an object is updated. In case of an update, the browser will reload and present the object.
Therefore an integrated BWS receiver does not have to support the "Refresh" parameter inside an `AdditionalHeader` parameter.

A.2.5 Object updates

MOT demands that the user application decoder is informed of object updates. In case of an integrated receiver, the MOT decoder could directly communicate with the browser and signal object updates.

A.3 PC based receivers

To assure that the browser will always contact the BWS server and not cache pages, it may be useful to include the http response parameters "*Cache-Control: no-cache*" and "*Pragma: no-cache*" in all http responses.

A.3.1 Mapping URLs to ContentNames

All object requests in http will have the form "*GET* RequestId HTTP-Version", for instance "GET /images/logo.jpg HTTP/1.1". Other http request header parameters will usually be provided in addition.

The requested URL path will be absolute, because the web browser will convert all partial links inside a web page to an absolute link. The absolute URL path will also always starts with a leading "/".

The absolute link is mapped to an MOT object (see clause A.1.1).

The error page (object not inside the current MOT directory) could use the http response header parameter "*Refresh*" that tells the browser to automatically reload the entry page of the BWS after some seconds; the page could also contain a link that references the entry page of the BWS.

The warning page (object not yet available) could also use the http response header parameter "*Refresh*" that tells the browser to automatically reload this page after some seconds. The same warning page will be returned for every reload request until the requested page is finally available or the page is removed from the MOT directory. In the latter case an error page (see above) will be returned.

A.3.2 The MOT parameter MimeType

The value of this parameter can be directly forwarded as the value of the http response header parameter "*Content-Type*".

A.3.3 The MOT parameter CompressionType

The MOT parameter `CompressionType` signals compression on transport level. One way to implement this parameter is to automatically decompress a compressed MOT body as soon as it is reassembled. The BWS server would then return the uncompressed MOT body to the browser.

However, if compression "gzip" is used, then an easier approach is possible:

The MOT decoder stores the compressed MOT bodies as they are received. When the browser requests a compressed object, the BWS server checks the object's header information and detects that the body is compressed. If the compression is "gzip", then the BWS server returns the compressed object to the browser and signals a "*Content-Encoding: gzip*" parameter in the http response header.

Since most browsers support "gzip" transport encoding, such an approach can be used to reduce memory in the MOT decoder's cache.

A.3.4 The MOT parameter AdditionalHeader

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.

The data field of this parameter is simply incorporated into the HTTP response header written in response to a URL request for the object. When including any specified additional header fields in an HTTP response header, such header fields should be the last header fields before the blank line that separates the HTTP message header from the HTTP message body.

The MOT parameter `AdditionalHeader` carries a data field without any terminating <CR><LF> sequence. However, HTTP demands that the <CR> <LF> sequence terminates all response header parameters. So this sequence must be added when creating the HTTP response header.

A.3.5 Object updates

MOT demands that the user application decoder is informed of object updates. The following steps are recommended for the BWS server to permit the browser to automatically reload updated objects. The coding of the HTTP parameters is described in RFC 2068 [6].

- When an object is returned to the browser, the BWS server first generates the http response header parameters such as "*Content-Type*", "*Transfer-Encoding*" and others (see above).
- In addition it will also generate the http response parameters "*Content-Length*" and "*Date*". The "*Content-Length*" parameter gives the length of the MOT body; the "*Date*" parameter should specify the date and time when the MOT body was completely reassembled (or stored to disk).
- To assure that the browser will check for an update, the BWS server will also add the http response parameter "*Refresh*" with the number of seconds until the browser should check for an update. If one of the MOT parameters `AdditionalHeader` of the MOT object contains the http response parameter "*Refresh*", then the BWS server shall return the value this MOT parameter and it shall not add an automatically generated "*Refresh*" parameter.
- The BWS server will then return the http response header together with the response body (the MOT body) to the browser.

After the time indicated by the http response parameter "*Refresh*", most browsers will issue a conditional http GET. This means that they will order the BWS server to return the object again unless its length and its date are unaltered. For this conditional http GET the browser needs to know the length and the date of every object is requests. That is the reason why the BWS server should generate "*Content-Length*" and "*Date*" parameters. It is not so important whether the value for "*Date*" is the date/time when the object was reassembled or stored to disk or another date/time. It is only important that the date/time of an object changes whenever a new version of the object is received.

If the BWS server detects that the browser already has the current version of an object (i.e. no update), it will not return the object again, but return a "304 Not modified". The browser thus knows that it does not have to re-display the object. This is not just a time-saver, but also assures that no redrawing of the display irritates the user. Most browsers will also scroll to the top of the page whenever the page is reloaded. Therefore a conditional http GET that avoids reloading (and redrawing) an unaltered page will significantly improve the user-friendliness.

NOTE: Unfortunately the above steps do not assure that an updated page will always automatically be reloaded and presented by the browser.
Many browsers for instance will periodically check if the HTML page was modified, but they do not necessarily also check if its inline images were modified. This even happens when the BWS server also returns the inline images with the necessary http response header.
Some browsers will also honour an http "*Refresh*" parameter when they load the page, but if the user presses the "Back" or "Forward" button and another page is presented (taken from the browser's cache), then the browser might no longer honour the http "*Refresh*" parameter of that page.

So the above means will increase the user-friendliness, but under some circumstances the user might still be forced to press the "Reload" button.

It is recommended to check multiple browsers to see if they can correctly handle the http response parameters. This way it is possible to tell which browsers are best suited for the BWS receiver.

History

Document history		
V1.1.1	August 2000	Publication
V2.1.1	January 2006	Publication