ITUPublications
Recommandations

Union internationale des télécommunications
Secteur des radiocommunications

Recommandation UIT-R BS.1352-4

(05/2023)

Série BS: Service de radiodiffusion sonore

Format des fichiers pour l'échange de programmes audio avec métadonnées sur supports informatiques

Avant-propos

Le rôle du Secteur des radiocommunications est d'assurer l'utilisation rationnelle, équitable, efficace et économique du spectre radioélectrique par tous les services de radiocommunication, y compris les services par satellite, et de procéder à des études pour toutes les gammes de fréquences, à partir desquelles les Recommandations seront élaborées et adoptées.

Les fonctions réglementaires et politiques du Secteur des radiocommunications sont remplies par les Conférences mondiales et régionales des radiocommunications et par les Assemblées des radiocommunications assistées par les Commissions d'études.

Politique en matière de droits de propriété intellectuelle (IPR)

La politique de l'UIT-R en matière de droits de propriété intellectuelle est décrite dans la «Politique commune de l'UIT-T, l'UIT-R, l'ISO et la CEI en matière de brevets», dont il est question dans la Résolution UIT-R 1. Les formulaires que les titulaires de brevets doivent utiliser pour soumettre les déclarations de brevet et d'octroi de licence sont accessibles à l'adresse http://www.itu.int/ITU-R/go/patents/fr, où l'on trouvera également les Lignes directrices pour la mise en oeuvre de la politique commune en matière de brevets de l'UIT-T, l'UIT-R, l'ISO et la CEI et la base de données en matière de brevets de l'UIT-R.

Séries des Recommandations UIT-R

(Egalement disponible en ligne: https://www.itu.int/publ/R-REC/fr)

Séries	Titre
BO	Diffusion par satellite
BR	Enregistrement pour la production, l'archivage et la diffusion; films pour la télévision
BS	**Service de radiodiffusion sonore**
BT	Service de radiodiffusion télévisuelle
F	Service fixe
M	Services mobile, de radiorepérage et d'amateur y compris les services par satellite associés
P	Propagation des ondes radioélectriques
RA	Radio astronomie
RS	Systèmes de télédétection
S	Service fixe par satellite
SA	Applications spatiales et météorologie
SF	Partage des fréquences et coordination entre les systèmes du service fixe par satellite et du service fixe
SM	Gestion du spectre
SNG	Reportage d'actualités par satellite
TF	Emissions de fréquences étalon et de signaux horaires
V	Vocabulaire et sujets associés

***Note**: Cette Recommandation UIT-R a été approuvée en anglais aux termes de la procédure détaillée dans la Résolution UIT-R 1.*

Publication électronique
Genève, 2024

RECOMMANDATION UIT-R BS.1352-4

Format des fichiers pour l'échange de programmes audio avec métadonnées sur supports informatiques

(Question UIT-R 34-3/6)

(1998-2001-2002-2007-2023)

Domaine d'application

La présente Recommandation contient la spécification du fragment d'extension audio pour la radiodiffusion[1] et son utilisation avec des données audio codées MIC ainsi que des données audio MPEG-1 ou MPEG-2 plus petites que 4 gigaoctets. On y trouvera également des informations de base sur le format RIFF et la façon dont il peut être étendu à d'autres types de données audio.

Mots clés

Fichier, format de fichier, onde, WAV, RIFF, BWF, fichier au format d'onde de radiodiffusion

L'Assemblée des radiocommunications de l'UIT,

considérant

a) que les supports d'enregistrement fondés sur l'informatique, y compris les disques et les bandes de données, ont pénétré dans tous les domaines de la production audio pour la radiodiffusion, à savoir l'édition non linéaire, la restitution à l'antenne et l'archivage;

b) que l'informatique offre de notables avantages en termes de souplesse d'exploitation, de flux de production et d'automatisation des stations et qu'elle est donc intéressante pour la modernisation des studios existants et pour la conception de nouvelles installations de studio;

c) que l'adoption d'un unique format de fichier pour l'échange de signaux simplifierait beaucoup l'interfonctionnement d'équipements individuels et de studios distants tout en facilitant l'intégration souhaitable des opérations d'édition, de restitution à l'antenne et d'archivage;

d) qu'un ensemble minimal d'informations relatives à la diffusion doit être inclus dans le fichier afin de décrire les métadonnées liées au signal audio;

e) que, pour assurer la compatibilité entre applications de complexités diverses, il faut adopter un ensemble minimal de fonctions communes à toutes les applications capables de traiter le format de fichier recommandé;

f) que la Recommandation UIT-R BS.646 définit le format audionumérique utilisé en production audio pour la diffusion radiophonique et télévisuelle;

g) que la nécessité d'échanger des données audio apparaîtra également lorsque les systèmes de codage selon l'ISO/CEI 11172-3 et 13818-3 seront utilisés pour comprimer le signal;

h) que la compatibilité avec les formats de fichiers actuellement disponibles sur le marché pourrait minimiser les efforts de l'industrie en vue de mettre en œuvre le présent format dans les équipements;

---

[1] Un fragment est le module de construction d'un fichier spécifié dans le format RIFF (Resource Interchange File Format) de Microsoft ®.

i) que, pour le champ «coding history», un format normalisé simplifierait l'utilisation de l'information après l'échange de fichiers de programme;

j) que la qualité d'un signal audio dépend du traitement appliqué à ce signal, notamment en cas de codage/décodage non linéaire lors de réductions du débit binaire,

reconnaissant

que le format de fichier défini dans la Recommandation UIT-R BS.2088 peut transmettre des données jusqu'à 16 exaoctets et prendre en charge les métadonnées relevant de la Recommandation BS.2076,

recommande

1 que, pour l'échange de programmes audio sur supports informatiques, les paramètres, la fréquence d'échantillonnage, la résolution de codage et la préaccentuation du signal audio soient déterminés conformément aux parties applicables de la Recommandation UIT-R BS.646;

2 que le format de fichier spécifié dans l'Annexe 1 soit utilisé pour l'échange de programmes audio en format modulation par impulsions et codage (MIC, en anglais: PCM) linéaire sur supports informatiques;

3 que, lorsque les signaux audio sont codés selon les systèmes ISO/CEI 11172-3 ou 13818-3, le format de fichier spécifié dans l'Annexe 1, complétée par l'Annexe 2, soit utilisé pour l'échange de programmes audio sur supports informatiques[2];

4 que, lorsque le format de fichier spécifié dans les Annexes 1 et/ou 2 est utilisé pour acheminer des informations sur des programmes audio traités sur une station de travail audionumérique, les métadonnées soient conformes aux spécifications détaillées fournies dans l'Annexe 3.

Annexe 1

Spécification du format d'onde de radiodiffusion

Format de diffusion des fichiers de données audio

1 Introduction

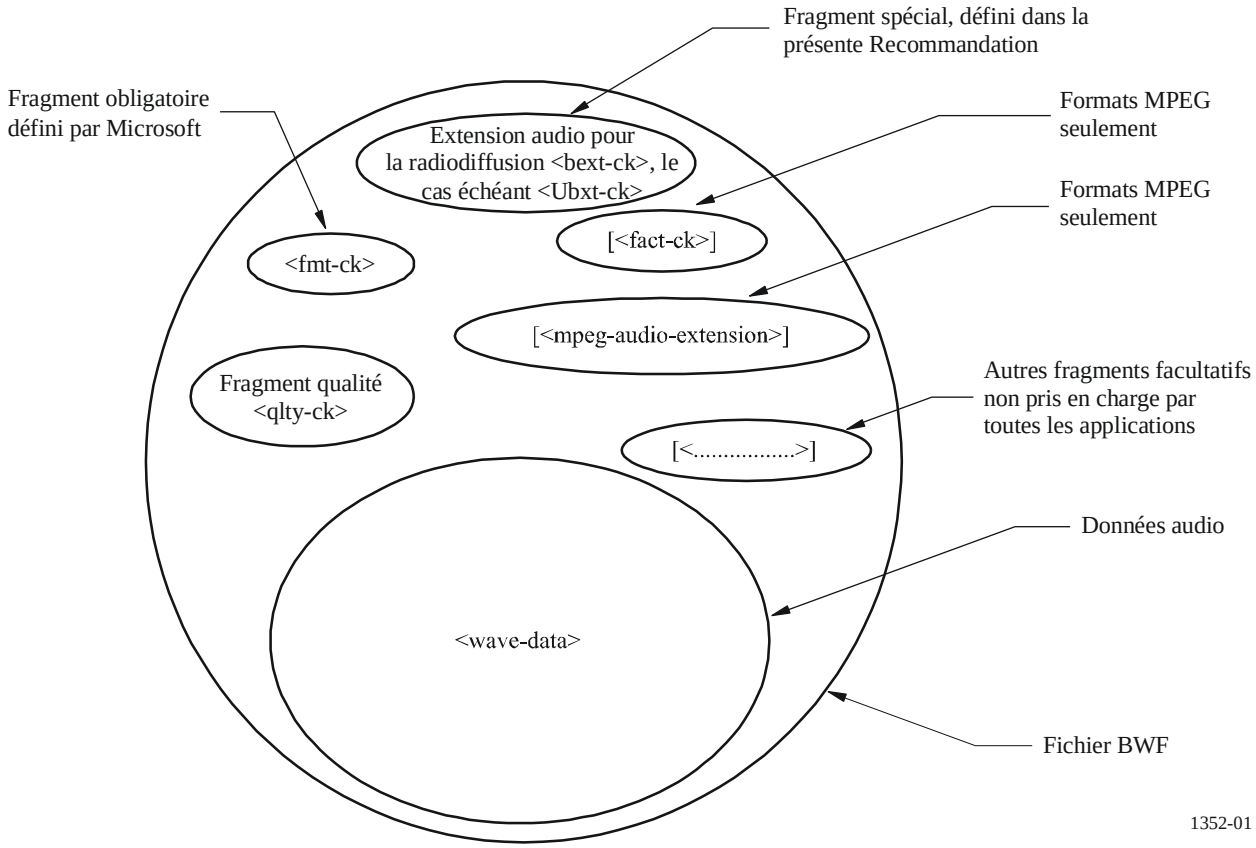
Le format d'onde de radiodiffusion (BWF, *broadcast wave format*) est fondé sur le fichier audio WAVE de Microsoft®, qui est d'un type spécifié dans le format RIFF (*Resource Interchange File Format*) de cette compagnie. Les fichiers WAVE contiennent spécifiquement des données audio. Le module de construction fondamental du format de fichier RIFF, appelé *fragment*, contient un groupe d'éléments informationnels étroitement associés. Il est composé d'un identificateur de fragment, d'une valeur d'entier représentant la longueur en octets et des informations. Un fichier au format RIFF se compose d'un ensemble de fragments.

Pour le format BWF, certaines restrictions sont appliquées au format WAVE original. De plus, le fichier BWF comporte un fragment particulier: <broadcast-audio-extension>, comme illustré sur la Fig. 1.

[2] Il est reconnu qu'une recommandation dans ce sens pourrait pénaliser les développeurs qui utilisent certaines plates-formes informatiques.

FIGURE 1

Fichier au format BWF



La présente Annexe spécifie le fragment d'extension audio pour la radiodiffusion qui est utilisé dans tous les fichiers BWF. Par ailleurs, la Pièce jointe 1 donne des renseignements sur le format RIFF de base et sur la façon dont il peut être étendu à d'autres types de données audio. Les détails du format audio à codage MIC sont également donnés dans la Pièce jointe 1. Les spécifications particulières de l'extension à d'autres types de données audio et les métadonnées sont incluses dans les Annexes 2 et 3 de la présente Recommandation.

1.1 Dispositions normatives

Le respect de la présente Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité). On considère donc que la Recommandation est respectée lorsque toutes ces dispositions sont observées.

Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe «devoir» ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

2 Fichiers au format BWF

2.1 Contenu d'un fichier BWF

Un fichier au format BWF doit commencer par l'en-tête «WAVE» du format RIFF de Microsoft® et doit comporter au moins les fragments suivants:

```
<WAVE-form>
RIFF('WAVE'
<fmt-ck>                                /*Format du signal audio:
                                        MIC/(MPEG)*/(Groupe d'experts pour les
                                        images animées)
<broadcast_audio_extension>             /*Information sur la séquence audio*/
<universal broadcast audio extension>   /*ubxt est requis uniquement pour la prise en
                                        charge du langage multi-octets*/
<fact-ck>                               /*Le fragment factuel est requis uniquement
                                        pour les formats MPEG*/
<mpeg_audio_extension>                  /*Le fragment «mpeg_audio_extension» est
                                        requis uniquement pour les formats MPEG*/
<wave-data> )                           /*Données audio*/
<quality-chunk>                         /*est requis uniquement lorsqu'il faut disposer
                                        d'informations sur des événements pertinents
                                        influant sur la qualité*/
```

NOTE – Des fragments additionnels peuvent être présents dans le fichier et certains d'eux peuvent sortir du cadre de la présente Recommandation. Les applications ne sont pas tenues d'interpréter ou d'utiliser ces fragments de sorte que l'intégrité des données contenues dans des fragments inconnus ne peut être garantie. Toutefois, les applications au format BWF doivent transmettre de manière transparente ces fragments inconnus.

2.2 Fragments déjà définis comme éléments de la norme RIFF

La norme RIFF est définie dans des documents publiés par Microsoft®[3] Corporation. La présente application utilise un certain nombre de fragments qui y sont déjà définis. Il s'agit des suivants:

– fmt-ck

– fact-ck

Les descriptions actuelles de ces fragments sont données pour information dans la Pièce jointe 1 de l'Annexe 1.

2.3 Fragment d'extension audio pour la radiodiffusion[4]

Les paramètres supplémentaires qui sont nécessaires pour l'échange de programmes entre radiodiffuseurs sont ajoutés dans un fragment spécifique «d'extension audio pour la radiodiffusion», défini comme suit:

```
broadcast_audio_extension typedef struct {
    DWORD   ckID;               /* (broadcastextension)ckID=bext. */
    DWORD   ckSize;             /* fragment de taille d'extension */
    BYTE    ckData[ckSize];     /* données du fragment */
}
typedef struct broadcast_audio_extension {
CHAR Description[256];          /* ASCII: «Description de la séquence sonore» */
CHAR Originator[32];            /* ASCII: «Nom de la source» */
```

[3] Le format RIFF (Resource Interchange File Format) de Microsoft est disponible (2005-12) à l'adresse http://www.tactilemedia.com/info/MCI_Control_Info.html.

[4] Pour la définition du fragment ubxt, voir le § 2.4; s'applique aux informations lisibles par l'homme du fragment text dans une série de caractères multi-octets.

```
CHAR OriginatorReference[32];        /* ASCII: «Référence de la source» */
CHAR OriginationDate[10];            /* ASCII: «yyyy:mm:dd» (année/mois/jour) */
CHAR OriginationTime[8];             /* ASCII: «hh:mm:ss» (heure, minute, seconde) */
DWORD TimeReferenceLow;              /*Premier comptage d'échantillons depuis minuit, mot
                                     faible */
DWORD TimeReferenceHigh;             /* Premier comptage d'échantillons depuis minuit, mot
                                     fort */
*/
WORD Version;                        /* Version du format BWF; nombre binaire non signé */
*/
BYTE UMID_0,                         /* Octet binaire 0 de SMPTE UMID */....
BYTE UMID_63,                        /* Octet binaire 63 de SMPTE UMID */
CHAR Reserved[190],                  /* 190 octets, réservés pour usage futur,
                                     mis à .NULL. *          /
CHAR CodingHistory[],                /* ASCII: «Chronologie des codages» */
} BROADCAST_EXT,
```

Champ	**Description**
Description	Chaîne de caractères ASCII (256 maximum) donnant une description libre de la séquence. Pour faciliter les applications qui n'affichent qu'une courte description, il est recommandé d'insérer un résumé de la description dans les 64 premiers caractères, les 192 caractères suivants étant utilisés pour les détails. Si la longueur de la chaîne est inférieure à 256 caractères, le dernier d'entre eux est suivi d'un caractère «néant». (0x00)
Originator	Chaîne de caractères ASCII (32 maximum) contenant le nom de la source/du producteur du fichier audio. Si la longueur de cette chaîne est inférieure à 32 caractères, le champ est terminé par un caractère «néant». (0x00)
OriginatorReference	Chaîne de caractères ASCII (32 maximum) donnant une référence univoque, attribuée par l'organisation source. Si la longueur de cette chaîne est inférieure à 32 caractères, le champ est terminé par un caractère «néant». (0x00) Un format normalisé pour l'identificateur de source «unique» (USID) à utiliser dans le champ OriginatorReference est décrit à la Pièce jointe 3 de l'Annexe 1.
OriginationDate	Chaîne de 10 caractères ASCII indiquant la date de la création de la séquence audio. Le format est «‘,année’, – ,‘mois,’ – ‘,jour,’» avec 4 caractères pour l'année et 2 caractères par autre élément. L'année est définie par un nombre compris entre 0000 et 9999. Le mois est défini par un nombre compris entre 1 et 12. Le jour est défini par un nombre compris entre 1 et 31.

	Le séparateur entre les éléments devrait être un trait d'union conformément à la norme ISO 8601. Il se peut que les caractères suivants: '_' soulignement, ':' deux points, ' ' espace, '.' point, soient utilisés dans des mises en œuvre précédentes. Il convient que l'appareil de lecture reconnaisse ces séparateurs.
OriginationTime	Chaîne de 8 caractères ASCII indiquant l'heure de création de la séquence audio. Le format est «',heure,' – ',minute,' – ',seconde,'» avec 2 caractères par élément. L'heure est définie par un nombre compris entre 0 et 23. La minute et la seconde sont définies par un nombre compris entre 0 et 59. Le séparateur entre les éléments devrait être un trait d'union conformément à la norme ISO 8601. Il se peut que les caractères suivants: '_' soulignement, ':' deux points, ' ' espace, '.' point, soient utilisés dans des mises en œuvre précédentes. Il convient que l'appareil de lecture reconnaisse ces séparateurs.
TimeReference	Ce champ contient le code temporel de la séquence. C'est une valeur codée sur 64 bits qui contient le premier comptage d'échantillons depuis minuit. Le nombre d'échantillons par seconde dépend de la fréquence d'échantillonnage, qui est définie dans le champ <nSamplesPerSec> issu du fragment de format <**fmt-ck**>.
Version	Nombre binaire non signé qui indique la version du fichier BWF. Pour la Version 1, ce nombre est mis à 0x0001.
UMID	Chaîne de 64 octets contenant un UMID étendu défini par la norme SMPTE 330M. Si un identificateur UMID de base de 32 octets est utilisé, les 32 derniers octets devraient être remplis de zéros. Si aucun identificateur UMID n'est disponible, les 64 octets devraient être remplis de zéros. NOTE – La longueur de l'identificateur UMID est codée dans l'en-tête de l'identificateur UMID proprement dit.
Reserved	Chaîne de 190 octets réservés pour une extension. Ces 190 octets devraient être mis à zéro.
CodingHistory	Bloc de taille variable de caractères ASCII comprenant zéro, une ou plusieurs chaînes terminées par la séance <CR><LF>. Le premier caractère non utilisé devrait être un caractère «néant» (0x00). Chaque chaîne donne une description d'un processus de codage appliqué aux données audio. Chaque nouvelle application de codage devrait ajouter une nouvelle chaîne contenant les informations appropriées. Un format normalisé de codage de ce champ est spécifié dans la Pièce jointe 2 de l'Annexe 1. Cette information doit indiquer le type de son (MIC ou MPEG), avec ses paramètres spécifiques: MIC: mode (mono, stéréo), longueur d'échantillon (8 ou 16 bits) et fréquence d'échantillonnage. MPEG: fréquence d'échantillonnage, débit binaire, couche (I ou II) et mode (mono, stéréo, stéréo mixte ou son bicanal).

Il est recommandé que les constructeurs de codeurs fournissent une chaîne ASCII à insérer dans le champ CodingHistory.

2.4 Fragment d'extension audio pour la radiodiffusion universelle

Les informations contenues dans le format d'extension audio pour la radiodiffusion (bext) défini au § 2.3 peuvent en outre être acheminées par un fragment spécialisé appelé «Extension audio pour la radiodiffusion universelle», ou fragment «ubxt» pour désigner les informations lisibles par l'homme du fragment bext dans des langages multi-octets. La structure de base de ce fragment de métadonnées est la même que pour le fragment bext. Quatre éléments lisibles par l'homme, uDescription, uOriginator, uOriginatorReference et uCodingHistory, sont décrits sous la forme de chaînes UTF-8 (*Format de transformation UCS à 8 bits*) au lieu de chaînes de caractères ASCII. Dans les trois premiers éléments, la taille des données est 8 fois supérieure à celle des éléments correspondants du fragment bext. La structure du fragment ubxt est définie comme suit:

```
typedef struct chunk_header {
        DWORD   ckID;                   /* (extension pour la radiodiffusion
                                           universelle)ckID=ubxt */
        DWORD   ckSize;                 /* taille du fragment d'extension */
        BYTE    ckData[ckSize];         /* données du fragment */
} CHUNK_HEADER;

typedef struct universal_broadcast_audio_extension {
        BYTE    uDescription[256*8];    /* UTF-8: «Description de la séquence sonore» */
        BYTE    uOriginator[32*8];      /* UTF-8: «Nom de la source» */
        BYTE    uOriginatorReference[32*8]; /* UTF-8: «Référence de la source» */
        CHAR    OriginationDate[10];    /* ASCII: «yyyy:mm:dd» (année/mois/jour) */
        CHAR    OriginationTime[8];     /* ASCII: «hh:mm:ss» (heure/minute/seconde) */
        DWORD   TimeReferenceLow;       /* Premier comptage d'échantillons depuis minuit, mot
                                           faible */
        DWORD   TimeReferenceHigh;      /* Premier comptage d'échantillons depuis minuit, mot
                                           fort */
        WORD    Version;                /* Version du format BWF; nombre binaire non signé */
        BYTE    UMID_0;                 /* Octet binaire 0 de SMPTE UMID */....
        BYTE    UMID_63;                /* Octet binaire 63 de SMPTE UMID */
        CHAR    Reserved[190];          /* 190 octets, réservés pour usage futur, mis à «NULL»
                                           */
        BYTE    uCodingHistory[];       /* UTF-8: «Chronologie des codages» */
} UNIV_BROADCAST_EXT;
```

Champ	Description
uDescription	Chaîne UTF-8, de 2 048 octets ou moins, donnant une description de la séquence. En l'absence de données disponibles ou si la longueur de cette chaîne est inférieure à 2 048 octets, le premier octet non utilisé doit être un caractère «néant» (0x00).

uOriginator	Chaîne UTF-8, de 256 octets ou moins, donnant le nom de la source du fichier audio. En l'absence de données disponibles ou si la longueur de cette chaîne est inférieure à 256 octets, le premier octet non utilisé doit être un caractère «néant» (0x00).
uOriginatorReference	Chaîne UTF-8, de 256 octets ou moins, donnant une référence attribuée par l'organisation source. En l'absence de données disponibles ou si la longueur de cette chaîne est inférieure à 256 octets, le premier octet non utilisé doit être un caractère «néant» (0x00).
OriginationDate	Chaîne de 10 caractères ASCII indiquant l'heure de création de la séquence audio. Le format est «‘,année’,-,’mois,’-‘,jour,’» avec 4 caractères pour l'année et 2 caractères par autre élément. L'année est définie par un nombre compris entre 0000 et 9999. Le mois est défini par un nombre compris entre 1 et 12. Le jour est défini par un nombre compris entre 1 et 31. Le séparateur placé entre les éléments devrait être un trait d'union conformément à la norme ISO 8601. Il se peut que les caractères suivants soient utilisés dans des mises en œuvre précédentes: ‘_’ soulignement, ‘:’ deux points, ‘ ’ espace, ‘.’ point. Il convient que l'appareil de lecture reconnaisse ces séparateurs.
OriginationTime	Chaîne de 8 caractères ASCII indiquant l'heure de création de la séquence audio. Le format est «‘heure,’-‘,minute,’-‘,seconde’» avec 2 caractères par élément. L'heure est définie par un nombre compris entre 0 et 23. La minute et la seconde sont définies par un nombre compris entre 0 et 59. Le séparateur placé entre les éléments devrait être un trait d'union conformément à la norme ISO 8601. Il se peut que les caractères suivants soient utilisés dans des mises en œuvre précédentes: ‘_’ soulignement, ‘:’ deux points, ‘ ’ espace, ‘.’ point. Il convient que l'appareil de lecture reconnaisse ces séparateurs.
TimeReference	Ce champ contient le code temporel de la séquence. C'est une valeur codée sur 64 bits qui contient le premier comptage d'échantillons depuis minuit. Le nombre d'échantillons par seconde dépend de la fréquence d'échantillonnage, qui est définie dans le champ <nSamplesPerSec> issu du fragment de format <**fmt-ck**>.
Version	Nombre binaire non signé qui indique la version du fichier BWF. Pour la version 1, ce nombre est mis à 0x0001.
UMID	Chaîne de 64 octets contenant un UMID étendu défini par la norme SMPTE 330M. Si un identificateur UMID de base à 32 octets est utilisé, les 32 derniers octets devraient être remplis de zéros. Si aucun identificateur UMID n'est disponible, les 64 octets devraient être remplis de zéros. NOTE – La longueur de l'identificateur UMID est codée dans l'en-tête de l'identificateur UMID proprement dit.
Reserved	Chaîne de 190 octets réservés pour une extension. Ces 190 octets devraient être mis à zéro.

uCoding History	Bloc de taille variable de caractères UTF-8 comprenant 0, une ou plusieurs chaînes terminées par la séquence <CR><LF>. Le premier octet non utilisé doit être un caractère «néant» (0x00). Chaque chaîne donne une description d'un processus de codage appliqué aux données audio. Chaque nouvelle application de codage devrait ajouter une nouvelle chaîne contenant les informations appropriées. Un format normalisé de codage de ce champ est spécifié dans la Pièce jointe 2 de l'Annexe 1. Cette information doit indiquer le type de son (MIC ou MPEG), avec ses paramètres spécifiques: MIC: mode (mono, stéréo), longueur d'échantillon (8 ou 16 bits) et fréquence d'échantillonnage. MPEG: fréquence d'échantillonnage, débit binaire, couche (I ou II) et mode (mono, stéréo, stéréo mixte ou son bicanal). Il est recommandé que les constructeurs de codeurs fournissent une chaîne ASCII à insérer dans le champ CodingHistory.

NOTE 1 – Tous les éléments, sauf uDescription, uOriginator, uOriginatorReference et uCodingHistory, doivent avoir le même contenu que celui de chaque élément correspondant du fragment bext (voir le § 2.3).

NOTE 2 – Lorsqu'une valeur de code donnée dans la séquence UTF-8 ne fait pas partie du sous-ensemble (défini au Chapitre 12 de la norme ISO/CEI 10646:2003), pris en charge par un élément de l'équipement de traitement, la valeur reste inchangée et n'est pas prise en compte pour le traitement.

Pièce jointe 1 de l'Annexe 1 (Informative)

Format RIFF des fichiers WAVE (.WAV)

Les informations contenues dans la présente la Pièce jointe sont extraites des spécifications du format de fichier RIFF de Microsoft®. Elles ne figurent ici qu'à titre d'information.

1 Format des fichiers audio de type WAVE

La forme WAVE est définie comme suit. Les programmes doivent attendre (et ignorer) tous les fragments inconnus qui seront rencontrés, comme avec toutes les formes de type RIFF. Le fragment <fmt-ck> doit cependant apparaître toujours avant le fragment <wave-data> et ces deux fragments sont obligatoires dans un fichier WAVE.

```
<WAVE-form> ->
        RIFF ( 'WAVE'
                <fmt-ck>            // Fragment de format
                [<fact-ck>]         // Fragment factuel
                [<other-ck>]        // Autres fragments facultatifs
                <wave-data> )       // Données audio
```

Les fragments WAVE sont décrits dans les paragraphes suivants:

1.1 Fragment de format WAVE

Le fragment de format WAVE <fmt-ck> spécifie le format des données audio <wave-data>. Il est défini comme suit:

```
<fmt-ck>    ->fmt( <common-fields>
                <format-specific-fields> )
<common-fields> ->
        struct{
                WORD wFormatTag;            /* Catégorie de format */
                WORD nChannels;             /* Nombre de canaux */
                DWORD nSamplesPerSec;       /* Fréquence d'échantillonnage */
                DWORD nAvgBytesPerSec;      /* Estimation du tampon */
                WORD nBlockAlign;           /* Longueur de bloc de données */
        }
```

Les champs contenus dans la portion <common-fields> du fragment sont les suivants:

Champ	Description
wFormatTag	Nombre indiquant la catégorie de format WAVE du fichier. Le contenu de la portion <format-specific-fields> du fragment <fmt-ck> ainsi que l'interprétation des données audio, dépendent de cette valeur.
nchannels	Nombre de canaux représentés dans les données audio, par exemple 1 pour la monophonie et 2 pour la stéréophonie.
nSamplesPerSec	Fréquence d'échantillonnage (en échantillons par seconde) à laquelle chaque canal doit être reproduit.
nAvgBytesPerSec	Nombre moyen d'octets de données audio à transférer par seconde. Au moyen de cette valeur, le logiciel de reproduction peut estimer la capacité tampon nécessaire.
nBlockAlign	Alignement en bloc (octets) des données audio. Le logiciel de reproduction a besoin de traiter un multiple de <nBlockAlign> octets de données à la fois, de sorte que la valeur de ce champ peut être utilisée pour l'alignement du tampon.

Le champ <format-specific-fields> se compose de zéro, un ou plusieurs octets de paramètres. Les paramètres insérés dépendent de la catégorie de format WAVE. On trouvera de plus amples informations dans les paragraphes qui suivent. Le logiciel de reproduction doit être écrit de façon à permettre (et à ignorer) tous paramètres inconnus apparaissant à la fin du champ <format-specific-fields>.

1.2 Catégories de format WAVE

La catégorie de format d'un fichier WAVE est spécifiée par la valeur du champ <wFormatTag> du fragment ‘fmt’. La représentation des données dans le champ <wave-data> et le contenu du champ <format-specific-fields> du fragment ‘fmt’ dépendent de la catégorie de format.

Les catégories ouvertes (non protégées) actuellement définies dans le format WAVE sont les suivantes:

wFormatTag	Valeur	Catégorie de format
WAVE_FORMAT_PCM	(0x0001)	Format MIC (modulation par impulsions et codage) de Microsoft®

WAVE_FORMAT_MPEG (0x0050) Audio MPEG-1 (audio seulement)

NOTE – Bien que d'autres formats WAVE aient été déposés par Microsoft®, seuls les formats ci-dessus sont actuellement utilisés pour les fichiers BWF. Le § 2 ci-après donne des détails sur le format WAVE de catégorie MIC. Le § 3 donne des informations générales sur d'autres formats WAVE. L'Annexe 2 donne des détails sur le format WAVE de catégorie MPEG. D'autres formats WAVE pourront être définis ultérieurement.

2 Format MIC

Si le champ <wFormatTag> du fragment <fmt-ck> est mis à la valeur WAVE_FORMAT_PCM, les données audio se composent d'échantillons représentés en format MIC. Pour les données audio à codage MIC, le champ <format-specific-fields> est défini comme suit:

```
<PCM-format-specific> ->
    struct{
            WORD nBitsPerSample;    /* Longueur d'échantillon */
    }
```

Le champ <nBitsPerSample> spécifie le nombre de bits de données utilisés pour représenter chaque échantillon dans chaque canal. S'il y a plusieurs canaux, la longueur d'échantillon est la même pour chaque canal.

Le champ <nBlockAlign> doit être égal à la formule suivante, après avoir été arrondi au plus proche entier:

$$\text{nchannels} \times \text{BytesPerSample}$$

La valeur de BytesPerSample doit être calculée en arrondissant nBitsPerSample au plus proche octet entier. Lorsque le mot de l'échantillon audio est inférieur à un nombre entier d'octets, les bits de poids fort de l'échantillon audio sont placés dans les bits de poids fort du mot de données et les bits de données non utilisés adjacents au bit de poids faible doivent être mis à zéro.

Pour les données à codage MIC, un champ <nAvgBytesPerSec> du fragment ‘fmt’ doit être égal à la formule suivante:

$$\text{nSamplesPerSec} \times \text{nBblockAlign}$$

NOTE – Conformément à la spécification initiale du format WAVE, il est possible, par exemple, de regrouper des échantillons à 20 bits provenant de 2 canaux en 5 octets qui se partageront un seul octet pour les bits de poids faible des deux canaux. La présente Recommandation spécifie un nombre entier d'octets par échantillon audio, afin de réduire toute ambiguïté dans les mises en œuvre et d'obtenir une compatibilité maximale dans l'échange de programmes.

2.1 Mise en paquet des données pour fichiers WAVE de catégorie MIC

Dans un fichier WAVE monophonique, les échantillons sont enregistrés consécutivement. Pour les fichiers WAVE stéréophoniques, le canal 0 représente le canal de gauche et le canal 1 le canal de droite. Dans les fichiers WAVE à canaux multiples (multicanaux), les échantillons sont entrelacés.

Les schémas suivants montrent la mise en paquets des données pour des fichiers WAVE mono et stéréo à 8 bits:

Mise en paquet des données pour MIC mono 8 bits

Échantillon 1	Échantillon 2	Échantillon 3	Échantillon 4
Canal 0	Canal 0	Canal 0	Canal 0

Mise en paquet des données pour MIC stéréo 8 bits

Échantillon 1		Échantillon 2	
Canal 0 (gauche)	Canal 1 (droite)	Canal 0 (gauche)	Canal 1 (droite)

Les schémas suivants montrent la mise en paquet des données pour fichiers WAVE mono et stéréo à 16 bits:

Mise en paquet des données pour MIC mono à 16 bits

Échantillon 1		Échantillon 2	
Canal 0 octet de poids faible	Canal 0 octet de poids fort	Canal 0 octet de poids faible	Canal 0 octet de poids fort

Mise en paquet des données pour MIC stéréo à 16 bits

Échantillon 1			
Canal 0 (gauche)	Canal 0 (gauche)	Canal 1 (droite)	Canal 1 (droite)
octet de poids faible	octet de poids fort	octet de poids faible	octet de poids fort

2.2 Format des données dans les échantillons

Chaque échantillon est contenu dans un nombre entier d'octets *i* dont la longueur est le plus petit nombre d'octets nécessaires. L'octet de poids faible est enregistré le premier. Les bits qui représentent l'amplitude d'échantillonnage sont enregistrés aux positions binaires de poids fort du nombre *i*, les autres bits étant mis à zéro.

Par exemple, si la longueur d'échantillon (indiquée par le champ <nBitsPerSample>) est de 12 bits, chaque échantillon est codé sur un entier de deux octets. Les quatre bits de poids faible du premier octet (de poids faible) sont mis à zéro. Le format des données et les valeurs extrêmes de diverses longueurs d'échantillons de forme d'onde MIC se présentent comme suit:

Longueur d'échantillon	**Format des données**	**Valeur maximale**	**Valeur minimale**
Un à huit bits	Entier non signé	255 (0xFF)	0
Au moins neuf bits	Entier signé i	Plus grande valeur positive de *i*	Plus grande valeur négative de i

Par exemple, les valeurs maximale, minimale et médiane des données de forme d'onde MIC codées sur 8 bits et sur 16 bits sont les suivantes:

Format	**Valeur maximale**	**Valeur minimale**	**Valeur médiane**
MIC 8-bits	255 (0xFF)	0	128 (0x80)
MIC 16-bits	32767(0x7FFF)	–32768(–0x8000)	0

2.3 Exemples de fichiers WAVE de type MIC

Exemple de fichier WAVE de type MIC avec fréquence d'échantillonnage de 11,025 kHz, mono, 8 bits par échantillon:

RIFF(‘WAVE’ fmt(1, 1, 11025, 11025, 1, 8)
data(<wave-data>))

Exemple de fichier WAVE de type MIC avec fréquence d'échantillonnage de 22,05 kHz, stéréo, 8 bits par échantillon:

RIFF('WAVE' fmt(1, 2, 22050, 44100, 2, 8)
data(<wave-data>))

Exemple de fichier WAVE de type MIC avec fréquence d'échantillonnage de 44,1 kHz, mono, 20 bits par échantillon:

RIFF('WAVE' INFO(INAM(«O Canada»Z))
fmt(1, 1, 44100, 132300, 3, 20)
data(<wave-data>))

2.4 Enregistrement des données de type WAVE

Le champ <**wave-data**> contient les données de forme d'onde sonore. Il est défini comme suit:

<wave-data> –> { <data-ck> }

<data-ck> –> data(<wave-data>)

2.5 Fragment factuel

Le fragment factuel <fact-ck> contient des informations importantes sur le contenu du fichier WAVE. Ce fragment est défini comme suit:

<fact-ck> –> fact(<dwFileSize:DWORD>) /* Nombre d'échantillons */

Ce fragment n'est pas requis pour les fichiers à modulation MIC.

Le fragment «factuel» sera élargi pour inclure toute autre information requise par de futurs formats WAVE. Les champs ajoutés seront placés après le champ <dwFileSize>. Les applications pourront utiliser le champ de dimension de fragment pour déterminer les champs présents.

2.6 Autres fragments facultatifs

Un certain nombre d'autres fragments sont spécifiés pour usage dans le format WAVE. Les détails de ces fragments sont donnés dans la spécification du format WAVE et dans ses mises à jour ultérieures.

NOTE – Le format WAVE peut prendre en charge d'autres fragments facultatifs qui peuvent être inclus dans des fichiers WAVE pour transporter des informations spécifiques. Comme indiqué dans la Note du § 2.1 de l'Annexe 1, ces fragments sont considérés, dans le format BWF, comme étant privés. Ils seront ignorés par les applications qui ne peuvent pas les interpréter.

3 Autres types de fichiers WAVE

Les informations suivantes ont été extraites des normes de données Microsoft®. Elles décrivent les extensions nécessaires des fichiers WAVE de base (utilisés pour l'audio à codage MIC) afin de traiter d'autres types de format WAVE.

3.1 Informations générales

Tous les types WAVE nouvellement définis doivent contenir à la fois un fragment factuel <fact-ck> et une description de format WAVE étendu, insérée dans le fragment de format <fmt-ck>. Les fichiers WAVE de format RIFF et de type WAVE_FORMAT_PCM n'ont pas besoin de comporter le fragment supplémentaire ni la description de format WAVE étendu.

3.2 Fragment factuel

Ce fragment contient des informations variables selon les fichiers, concernant le contenu du fichier WAVE. Il spécifie actuellement la longueur du fichier, en nombre d'échantillons.

Extension du format WAVE

La structure étendue du format WAVE, ajoutée au fragment <**fmt-ck**>, est utilisée pour définir toutes les données audio de format autre que MIC. Elle est décrite comme suit. La structure étendue du format WAVE général est utilisée pour tous les formats autres que MIC.

```
typedef struct waveformat_extended_tag {
    WORD    wFormatTag,         /* type de format */
    WORD    nChannels,          /* nombre de voies (c'est-à-dire mono, stéréo...) */
    DWORD   nSamplesPerSec,     /* fréquence d'échantillonnage */
    DWORD   nAvgBytesPerSec,    /* pour l'estimation du tampon */
    WORD    nBlockAlign,        /* longueur de bloc de données */
    WORD    wBitsPerSample,     /* nombre de bits par échantillon de données mono */
    WORD    cbSize,             /* comptage d'octets de la forme étendue */
} WAVEFORMATEX;
```

Champ	**Description**
wFormatTag	Ce champ définit le type de fichier WAVE.
nChannels	Nombre de voies représentées dans les données audio: 1 pour la monophonie et 2 pour la stéréophonie.
nSamplesPerSec	Fréquence d'échantillonnage du fichier WAVE. La valeur de ce champ doit être 48 000 ou 44 100, etc. Cette fréquence est également utilisée par le champ de longueur d'échantillon contenu dans le fragment factuel afin de déterminer la durée des données.
nAvgBytesPerSec	Nombre moyen d'octets de données audio à transférer par seconde. Au moyen de la valeur <**nAvgBytesPerSec**>, le logiciel de reproduction peut estimer la capacité tampon nécessaire.
nBlockAlign	Alignement en bloc (octets) des données audio dans le fragment <**data-ck**>. Le logiciel de reproduction a besoin de traiter un multiple de <**nBlockAlign**> octets de données à la fois, de sorte que la valeur de ce champ peut être utilisée pour l'alignement du tampon.
wBitsPerSample	Nombre de bits par échantillon et par voie. Chaque voie est censée avoir la même résolution de codage des échantillons. Si ce champ n'est pas nécessaire, il doit être mis à zéro.
cbSize	Longueur en octets des informations supplémentaires contenues dans l'en-tête de format WAVE, à l'exclusion de la longueur de la structure d'extension WAVEFORMATEX.

NOTE – Les champs qui suivent le champ <cbSize> contiennent des informations spécifiques nécessaires pour le format défini dans le champ <wFormatTag>. Tous les formats WAVE qui peuvent être utilisés dans le format BWF seront spécifiés dans des Suppléments particuliers de la présente Recommandation.

Pièce jointe 2 de l'Annexe 1 (Informative)

Spécification du format du champ <CodingHistory>

Introduction

Le champ <CodingHistory> du fragment <bext> est, par définition, un ensemble de chaînes contenant la chronologie des processus de codage. Une nouvelle ligne doit être ajoutée à chaque modification de la chronologie de codage. Chaque ligne doit contenir une chaîne variable pour chaque paramètre de codage. La commande CR/LF doit figurer à la fin de chaque ligne. Un format de chaîne de chronologie de codage est indiqué ci-après.

Syntaxe

Les lignes se conformeront à la syntaxe suivante:

Paramètre	Chaîne variable <allowed option>
Algorithme de codage	A=<ANALOGUE, PCM, MPEG1L1, MPEG1L2, MPEG1L3, MPEG2L1, MPEG2L2, MPEG2L3>
Fréquence d'échantillonnage (Hz)	F=<16000,22050,24000,32000,44100,48000>
Débit binaire (kbit/s par canal)	B=<toute valeur de débit autorisée dans la norme MPEG-2 (ISO/CEI 13818-3)>
Longueur de mot (bits)	W=<8, 12, 14, 16, 18, 20, 22, 24>
Mode	M=<mono, stéréo, mono double, stéréo combiné>
Texte, chaîne libre	T=<chaîne texte ASCII libre pour utilisation domestique. La chaîne ne doit pas comporter de virgules (ASCII 2Chex). Exemples de contenus: numéro d'identification; type de codec; type analogique/numérique>

Les chaînes variables doivent être séparées par des virgules (ASCII $2C_{hex}$). La commande CR/LF doit figurer à la fin de chaque ligne.

Variable `B` = utilisée seulement pour le codage MPEG.

Variable `W` = pour codage MPEG, doit être utilisée pour indiquer la longueur de mot à l'entrée PCM du codeur MPEG.

Exemples de champs de chronologie de codage

Exemple 1

A=PCM,F=48000,W=16,M=stéréo,T=original,CR/LF

A=MPEG1L2,F=48000,B=192,W=16,M=stéréo,T=PCX9,CR/LF

Interprétation de l'exemple 1

Ligne 1

Le fichier d'origine est enregistré sous forme de fichier BWF linéaire, avec codage MIC présentant les caractéristiques suivantes:

– Fréquence d'échantillonnage: 48 kHz
– Résolution de codage: 16 bits par échantillon
– Mode: stéréo
– Statut: codage d'origine

Ligne 2

Le fichier d'origine a été converti en fichier BWF MPEG-1 couche II avec les paramètres suivants:

– Fréquence d'échantillonnage: 48 kHz
– Débit par canal: 192 kbit/s
– Résolution de codage: 16 bits
– Mode: stéréo
– Codeur: PCX9 (Digigram)

Exemple 2 – numérisation d'un signal analogique

A=ANALOGUE,M=stereo,T=StuderA816; SN1007; 38; Agfa_PER528,<CR/LF>

A=PCM,F=48000,W=18,M=stereo,T=NVision; NV1000; A/D,<CR/LF>

A=PCM,F=48000,W=16,M=stereo,T=PCX9;DIO,<CR/LF>

Interprétation de l'exemple 2

Ligne 1

La bande magnétique analogique Agfa PER528 a été lue sur un magnétophone Studer A816, numéro de série 1007:

– Vitesse de bande: 38 cm/s
– Mode: stéréo

Ligne 2

L'enregistrement a été numérisé à l'aide d'un convertisseur analogique/numérique de type NVision NV1000 présentant les caractéristiques suivantes:

– Fréquence d'échantillonnage: 48 kHz
– Résolution de codage: 18 bits par échantillon
– Mode: stéréo

Ligne 3

L'enregistrement a été mis en mémoire sous forme de fichier BWF, avec codage MIC linéaire à l'entrée numérique d'une carte d'interface PCX9 présentant les caractéristiques suivantes:

– Fréquence d'échantillonnage: 48 kHz
– Résolution de codage: 16 bits par échantillon
– Mode: stéréo

Pièce jointe 3
de l'Annexe 1
(Informative)

Spécification du format pour l'identificateur de source «unique» (USID) à utiliser dans le champ <OriginatorReference>

USID

L'identificateur USID dans le champ <OriginatorReference> est généré à l'aide de plusieurs sources de randomisation indépendantes en vue de garantir son caractère unique en l'absence d'une autorité d'attribution. La méthode de randomisation est simple et efficace car elle combine des éléments spécifiques (utilisateur, machine et heure plus un numéro aléatoire), à savoir:

CC	Code pays: (2 caractères) fondé sur la norme ISO 3166[5] [ISO, 1997].
OOOO	Code organisation: 4 caractères.
NNNNNNNNNNNN	Numéro de série: (12 caractères extraits du modèle d'enregistreur et du numéro de série). Ce numéro devrait permettre d'identifier le type de machine et le numéro de série.
HHMMSS	OriginationTime: (6 caractères) extrait du champ <OriginationTime> du format BWF.

Ces éléments devraient être suffisants pour identifier un enregistrement donné sous une forme utilisable ainsi que d'autres sources d'information, formelles ou informelles.

De plus, l'identificateur USID contient l'élément suivant:

RRRRRRRR	Numéro aléatoire (8 caractères) généré localement par l'enregistreur à l'aide d'un algorithme suffisamment aléatoire.

Cet élément sert à identifier séparément les fichiers comme les voies stéréo ou les pistes dans des enregistrements multipistes, qui sont établis en même temps.

Exemples d'identificateur USID

Exemple 1

Indicateur USID généré par un appareil Tascam DA88, S/N 396FG347A, exploité par la RAI, Radiotelevisione Italiana, à: 12:53:24

```
Format UDI:  CCOOOONNNNNNNNNNNNHHMMSSRRRRRRRR
Exemple UDI: ITRAI0DA88396FG34712532498748726
```

Exemple 2

Identificateur USID généré par un xxxxxxx, S/N sssssssss, exploité par YLE, Finnish Broadcasting, à: 08:14:48

```
Format UDI:  CCOOOONNNNNNNNNNNNHHMMSSRRRRRRRR
Exemple UID: FIYLE0xxxxxxssssss08144887724864
```

[5] ISO 3166-1:1997 Codes pour la représentation des noms de pays et de leurs subdivisions – Partie 1: codes pays (voir: http://www.din.de/gremien/nas/nabd/iso3166ma/index.html).

Pièce jointe 4
de l'Annexe 1
(Informative)

Définition d'un fragment Niveau d'enveloppe de crête facultatif <levl-ck> pour le format BWF

L'échange de fichiers audio entre des postes de travail peut accélérer le lancement, l'affichage et le traitement d'un fichier si des données sur les niveaux du signal audio de crête sont disponibles dans le fichier. L'adjonction d'un fragment <***levl***> dans un fichier au format d'onde de radiodiffusion (BWF) [1] permet la mise en place d'une norme pour l'enregistrement et le transfert de données sur les crêtes du signal obtenues par sous-échantillonnage des données audio. On peut utiliser ces données présentes dans le fragment pour obtenir l'enveloppe des caractéristiques audio du fichier. Une application audio pourra ainsi afficher rapidement les fichiers audio sans perte de précision trop importante.

De plus, il est possible d'envoyer la crête des crêtes, *premier* échantillon audio dont la valeur absolue est la valeur maximale de la totalité du fichier audio. Une application audio peut utiliser cette information pour normaliser un fichier en temps réel sans devoir numériser tout le fichier (ce qui a déjà été fait par l'expéditeur).

1 Terminologie

Le signal audio est divisé en blocs. Une **trame crête** est générée ici pour chaque bloc audio. Il existe n **valeurs de crête** pour chaque trame crête, n étant le nombre de canaux de crête. Chaque valeur de crête peut comprendre un **point de crête** (positif seulement) ou deux (un positif et un négatif).

1.1 Génération de valeurs de crête

Le signal audio est divisé en blocs d'échantillons de longueur constante. La longueur par défaut et recommandée des blocs est de 256 échantillons par canal.

On évalue les échantillons de chaque canal pour trouver les points de crête (valeurs maximales). Il est recommandé de trouver des points de crête distincts pour les échantillons positifs et négatifs mais seule la valeur absolue (positive ou négative) peut être utilisée. Tous les points de crête sont des valeurs non signées.

Les points de crête sont arrondis à l'un des deux formats, 8 ou 16 bits. Dans la plupart des cas, le format à 8 bits suffit. Le format à 16 bits devrait s'appliquer aux cas qui nécessitent une plus grande précision.

Les points de crête formatés pour chaque canal sont assemblés en trames crête. Chaque trame crête contient les points de crête positifs et négatifs (ou le point de crête absolu) pour chaque canal selon le même ordre que pour les échantillons audio.

Ces trames crête sont acheminées de la même manière que les données dans le fragment d'enveloppe de crête. Ce fragment d'enveloppe de crête débute par un en-tête qui contient les informations permettant d'interpréter les données de crête.

La **crête des crêtes** est le *premier* échantillon audio dont la valeur absolue est la valeur maximale de la totalité du fichier audio. Plutôt que d'enregistrer la crête des crêtes comme valeur d'échantillon, on enregistre sa *position*. Autrement dit, on enregistre un indice de trame de l'échantillon audio. Chaque application sait donc *où* lire la crête des crêtes dans le fichier audio. Il serait plus difficile d'enregistrer une valeur de crête car elle dépend du format binaire des échantillons audio (nombres entiers, à virgules flottantes, à deux chiffres ...).

NOTES:

– L'en-tête utilise uniquement des DWORD (valeurs de 4 octets) ou des multiples de 4 octets pour éviter les problèmes d'alignement des structures dans différents compilateurs.

– La longueur totale de l'en-tête est de 128 octets, afin d'éviter le mauvais alignement du cache.

2 Fragment d'enveloppe de crête

Le fragment d'enveloppe de crête, <***levl***>, comprend un en-tête suivi des données des points de crête. La longueur totale du fragment sera variable, en fonction du contenu audio, de la longueur du bloc et du formatage des données de crête.

```
typedef struct peak_envelope
{
        CHAR        ckID[4],            /* {'l','e','v','l'} */
        DWORD       ckSize,             /* longueur du fragment */
        DWORD       dwVersion,          /* informations concernant la version */
        DWORD       dwFormat,;          /* format d'un point de crête */
                                        1 = caractère non signé
                                        2 = entier court non signé
        DWORD       dwPointsPerValue,   /* 1 = point de crête positif seulement
                                        2 = points de crête positif ET négatif */
        DWORD       dwBlockSize,        /* trames par valeur */
        DWORD       dwPeakChannels,     /* nombre de canaux */
        DWORD       dwNumPeakFrames,    /* nombre de trames crête */
        DWORD       dwPosPeakOfPeaks,   /* indice de trame de l'échantillon audio /* ou
                                        0xFFFFFFFF si inconnu */
        DWORD       dwOffsetToPeaks,    /* doit généralement être égal à la longueur de cet
                                        en-tête, mais pourrait aussi avoir une valeur plus
                                        élevée */
        CHAR        strTimestamp[28],   /* ASCII: indication horaire des données de crête */
```

2.1 Élément du fragment «levl»

ckID Tableau de 4 caractères {«l», «e», «v», «l»}[6] pour l'identification du fragment.

ckSize Longueur du reste du fragment (ne tient pas compte des 8 octets utilisés par ckID et ckSize.)

dwVersion Version du fragment peak_envelope. Commence par 0000.

dwFormat Format des données de l'enveloppe de crête. Deux formats sont autorisés[7]:

---

[6] La définition de DWORD ckID = «levl» ne serait pas unique. Différents compilateurs C produisent différents ordres de présentation des caractères. Nous avons donc défini le caractère ckID[4] = {«l», «e», «v», «l»}.

[7] Sachant que toute application audio qui prend en charge le fragment **«levl»** devrait mettre en oeuvre tous les formats possibles, seuls deux formats sont autorisés. Dans la plupart des cas, le format de caractère non signé (8 bits) suffit. Le format court non signé (16 bits) devrait s'appliquer aux cas qui nécessitent une plus grande précision.

dwFormat	Valeur	Description
LEVL_FORMAT_UINT8	1	caractère non signé pour chaque point de crête
LEVL_FORMAT_UINT16	2	nombre entier court non signé pour chaque point de crête

dwPointsPerValue Indique le nombre de points de crête par valeur de crête, à savoir, un ou deux.

dwPointsPerValue = 1

Chaque valeur de crête se compose d'un point de crête, valeur maximale des valeurs absolues des échantillons audio **dwBlockSize** dans chaque bloc:

$$\max\{abs(X1),...,abs(Xn)\}$$

NOTE – Dans ce cas, la forme d'onde affichée sera toujours symétrique par rapport à l'axe horizontal.

dwPointsPerValue = 2

Chaque valeur de crête se compose de deux points de crête. Le premier correspond à la valeur *positive* la plus élevée des échantillons audio **dwBlockSize** du bloc. Le second correspond à la crête *négative* des échantillons audio **dwBlockSize** du bloc.

Il est recommandé d'utiliser deux points de crête (**dwPointsPerValue = 2)** car les formes d'onde non symétriques (par exemple, un décalage DC) seront correctement affichées.

dwBlockSize Nombre d'échantillons audio utilisés pour générer chaque trame crête. Ce nombre est variable. La longueur du bloc par défaut et recommandée est de 256 échantillons.

dwPeakChannels Nombre de canaux de crête[8].

dwNumPeakFrames Nombre de trames crête ou nombre entier obtenu par arrondi vers le bas dans le calcul suivant:

$$\text{dwNumPeakFrames} = \frac{(\text{numAudioFrame} + \text{dwBlockSize})}{\text{dwBlockSize}}$$

ou par arrondi vers le haut dans le calcul suivant:

$$\text{dwNumPeakFrames} = \frac{\text{numAudioFrame}}{\text{dwBlockSize}}$$

Où numAudioFrame est le nombre d'échantillons audio dans chaque canal des données audio.

Par exemple, pour un rapport de crête (longueur de bloc) de 256, cela signifie:

0 échantillon audio -> 0 trame crête
1 échantillon audio -> 1 trame crête
256 échantillons audio -> 1 trame crête
257 échantillons audio -> 2 trames crête
7582 échantillons audio -> 30 trames crête.

[8] En général, le nombre de canaux de crête est égal au nombre de canaux audio. Si ce nombre est égal à un, la même forme d'onde sera affichée pour chaque canal audio.

dwPosPeakOfPeaks Une application audio peut utiliser cette information pour normaliser un fichier sans devoir numériser tout le fichier (ce qui a déjà été fait par l'expéditeur). Cela permet d'améliorer la performance et aussi de normaliser un fichier en temps réel.

La **crête des crêtes** est le *premier* échantillon audio dont la valeur absolue est la valeur maximale de la totalité du fichier audio.

Plutôt que d'enregistrer la crête des crêtes comme valeur d'échantillon, on enregistre sa *position*. Autrement dit, on enregistre un indice de trame de l'échantillon audio. Chaque application sait donc *où* lire la crête des crêtes dans le fichier audio. Il serait plus difficile d'enregistrer une valeur de crête car elle dépend du format binaire des échantillons audio (nombres entiers, à virgules flottantes, à deux chiffres).

Si la valeur est `0xFFFFFFFF`, cela signifie alors que la crête des crêtes n'est pas connue.

dwOffsetToPeaks Décalage des données de crête par rapport au début de l'en-tête. Est généralement égal à la longueur de l'en-tête mais pourrait avoir une valeur plus élevée. Peut être utilisé pour garantir que les données de crête commencent à la limite `DWORD`.

strTimeStamp Chaîne contenant l'indication horaire de la création des données de crête. Elle est formatée comme suit[9]:

«YYYY:MM:DD:hh:mm:ss:uuu»

où:

YYYY:	année
MM:	mois
DD:	jour
hh:	heures
mm:	minutes
ss:	secondes
uuu:	millisecondes

Exemple: «2000:08:24:13:55:40:967»

2.2 Format d'un point de crête

Une valeur de crête se compose d'un ou deux points de crête, signalée par **dwPointsPerValue**. Le fanion **dwFormat** indique le format des nombres représentant les points de crête dans chaque trame crête.

[9] L'avantage de ce format est qu'il n'est pas limité dans le temps et qu'il est facile à lire. (D'autres formats utilisent un DWORD pour désigner les secondes depuis 1970; la limite sera atteinte dans près de 125 ans.)

		dwPointsPerValue	
		= 1	**= 2**
dwFormat		Le nombre correspond à la crête absolue	Le premier nombre correspond à la crête positive Le second nombre correspond à la crête négative (À noter que la crête «négative» est enregistrée comme nombre «positif»)
= 1	**levl_format_uint8**	*caractère non signé (0...255)*	*caractère non signé (0...255)* *caractère non signé (0...255)*
= 2	**levl_format_uint16**	*entier court non signé* (0...65535)	*entier court non signé* (0...65535) *entier court non signé* (0...65535)

2.3 Fichiers de crête multicanaux

S'agissant des fichiers audio multicanaux, les valeurs de crête uniques de chaque canal sont entrelacées. Un ensemble de valeurs de crête entrelacées est appelé trame crête. L'ordre des valeurs de crête à l'intérieur d'une trame crête correspond à la position des points d'échantillon à l'intérieur de la trame de données audio au format RIFF.

2.4 Synchronisation avec le fichier audio

Le fichier de crête doit être reconstitué si l'une des deux conditions ci-après est remplie:

– L'indication horaire est antérieure à celle du fichier audio.

– Le nombre de trames crête ne correspond pas au nombre de trames d'échantillons dans le fichier audio.

2.5 Ordre des octets

Étant donné que le fichier au format d'onde de radiodiffusion (BWF) est une extension du format RIFF, tous les numéros sont enregistrés en commençant par le bit de plus faible poids.

Pièce jointe 5 de l'Annexe 1 (Informative)

Définition d'un fragment Liaison facultatif <link-ck> pour le format BWF

Introduction

Le fichier au format d'onde de radiodiffusion (BWF) autorise une longueur maximale de 4 gigaoctets même si dans la pratique de nombreuses applications RIFF/Wave n'admettront qu'une longueur maximale de 2 gigaoctets. Pour les données audio qui dépassent ces limites, il faut scinder l'information audio en plusieurs fichiers BWF. Le fragment <link> fournit des données de connexion avec une sortie audio homogène répartie sur plusieurs fichiers.

1 Terminologie

File-set Ensemble de fichiers interconnectés appartenant à un signal audio continu.

Filename Noms attribués à chaque fichier dans l'ensemble de fichiers.

Liste de fichiers Liste de noms de fichiers dans l'ensemble de fichiers.

Attribut «actuel» Attribut signalant le nom de fichier dans la liste de fichiers comme étant le fichier en cours (ou «actuel»). Tous les autres noms de fichiers de la liste de fichiers sont désignés par «autre».

Identificateur de fichier Identificateur facultatif qui devrait être le même pour tous les fichiers d'un ensemble de fichiers.

Élément «privé» Élément additionnel du fragment pour enregistrer des informations d'ordre privé dans la liste de fichiers.

Fragment <link> Fragment contenu dans tous les fichiers d'un ensemble de fichiers. Il contient un en-tête suivi d'une liste de fichiers et, à titre facultatif, un identificateur de fichier et un élément «privé». Les données présentes dans le fragment sont enregistrées en format XML 1.0[10], format courant pour l'échange de données.

2 Structure du fragment liaison

2.1 Aperçu général

Le fragment <link> se compose d'un en-tête suivi de l'information de connexion enregistrée en format XML (langage de balisage extensible). La longueur totale du fragment sera variable.

```
typedef struct link
{CHAR   CkID[4],        /* {'l','i','n','k'} */
        DWORD   CkSize,         /* taille du fragment */
        CHAR    XmlData[ ],     /* information de connexion en format XML */
    }
Link_chunk,
```

Field **Description** **ckID** Tableau de 4 caractères {‘l‘, ‘i‘, ‘n‘, ‘k‘}[11] pour l'identification du fragment.

CkSize Longueur de la section de données du fragment (sans les 8 octets utilisés par ckID et ckSize.)

XmlData Cette mémoire tampon contient l'information de connexion en format XML (caractères ASCII).

---

[10] Langage de balisage extensible (XML) Recommandation 1.0 W3C, 10 février 1998 http://www.w3.org/TR/1998/REC-xml-19980210.

[11] La définition DWORD ckID = «link» ne serait pas unique. Différents compilateurs C produisent différents ordres de présentation des caractères. Nous avons donc défini le caractère ckID[4] = {‘l‘, ‘i‘, ‘n‘, ‘k‘}.

2.2 Structure de données XML dans le champ de données variable <xmlData>

La structure de données est hiérarchique. Les données sont enregistrées dans des chaînes de texte. Pour la spécification exacte de la syntaxe, un document de transfert de données (DTD) est ajouté.

```
<LINK>
        <FILE type="…">
        <FILENUMBER>...</FILENUMBER>
        <FILENAME>...</FILENAME>
        </FILE>
        .......
        Possible further FILE elements
        .......
        <ID>...</ID>        optional
        <PRIVATE>           optional
        ..... implementation dependent
        </PRIVATE>
</LINK>
```

LINK Élément racine des données XML. L'élément LINK contient un ou plusieurs éléments FILE avec la description du fichier. Il peut aussi contenir un identificateur ID et/ou un élément PRIVATE.

ID L'identificateur ID est commun à tous les fichiers d'un ensemble de fichiers donné. Il est enregistré comme chaîne de caractères possible conformément à la définition #PCDATA de la spécification XML 1.0, qui comprend tous les caractères visibles, espaces ASCII, etc.

PRIVATE L'élément PRIVATE peut contenir des informations dépendant de la mise en œuvre qui regroupent n'importe quel type de données XML (comme d'autres éléments ou #PCDATA).

FILE L'élément FILE contient l'élément FILENUMBER et l'élément FILENAME. L'attribut type devrait être «actuel» si le fichier de la liste décrit le fichier auquel appartient le fragment. Tous les autres fichiers devraient avoir l'attribut de type «autre».

Le nom du fichier devrait être le même que celui qui apparaît dans la liste des fichiers.

FILENUMBER Les fichiers devraient être numérotés de manière séquentielle en fonction de leur ordre chronologique dans l'ensemble de fichiers. Il convient d'utiliser des nombres entiers (caractères ASCII) commençant par le chiffre 1.

FILENAME Chaîne de caractères enregistrée dans le même format que l'identificateur ID.

2.3 Définition du type de document pour la structure XML du fragment <link>

La définition du type de document (DTD) est décrite dans la spécification XML 1.0 comme définition de la syntaxe d'une structure XML. Le format et les attributs des différents éléments du fragment <link> sont décrits ci-après, y compris les sous-éléments et leur multiplicité.

L'élément LINK devrait contenir un ou plusieurs sous-éléments FILE («+» indique un ou plus); il peut contenir un sous-élément ID ainsi qu'un sous-élément PRIVATE («?» indique un ou aucun).

Chaque élément FILE devrait contenir un sous-élément FILENUMBER et un sous-élément FILENAME. Il convient de spécifier un attribut de type, qui peut être «actuel» ou «autre».

Les sous-éléments FILENUMBER, FILENAME et ID doivent contenir des chaînes de caractères (appelées #PCDATA en format XML).

Le sous-élément PRIVATE peut contenir n'importe lequel des éléments définis. Si le sous-élément PRIVATE doit contenir des éléments autres que les éléments définis, la définition DTD doit être modifiée en conséquence.

```
<!ELEMENT LINK          (FILE+, ID?, PRIVATE?)>
<!ELEMENT FILE          (FILENUMBER, FILENAME)>
<!ATTLIST FILE          type («actual» | «other») #REQUIRED>
<!ELEMENT FILE          NUMBER (#PCDATA)>
<!ELEMENT FILE          NAME (#PCDATA)>
<!ELEMENT ID            (#PCDATA)>
<!ELEMENT PRIVATE       ANY>
```

3 Renommage de fichiers interconnectés

Si un ou plusieurs noms de fichier sont modifiés, il convient de modifier les entrées FILENAME correspondantes dans chacun des fragments <link> appartenant à l'ensemble des fichiers dans sa globalité.

Dans cet exemple, le signal sonore continu a été scindé en un ensemble de fichiers comprenant trois fichiers BWF appelés «Sinatra_1.wav», «Sinatra_2.wav», et «Sinatra_3.wav». Les structures XML des fragments <link> des trois fichiers sont identiques, sauf l'attribut de type.

3.1 Fragment <link> de «Sinatra_1.wav»

```
<LINK>
        <FILE type=«actual»>
                <FILENUMBER>1</FILENUMBER>
                <FILENAME>Sinatra_1.wav</FILENAME>
        </FILE>
        <FILE type=«other»>
                <FILENUMBER>2</FILENUMBER>
                <FILENAME>Sinatra_2.wav</FILENAME>
        </FILE>
        <FILE type=«other»>
                <FILENUMBER>3</FILENUMBER>
                <FILENAME>Sinatra_3.wav</FILENAME>
        </FILE>
        <ID>73365869</ID>
</LINK>
```

3.2 Fragment <link> de «Sinatra_2.wav»

```
<LINK>
        <FILE type=«other»>
                <FILENUMBER>1</FILENUMBER>
                <FILENAME>Sinatra_1.wav</FILENAME>
        </FILE>
        <FILE type=«actual»>
                <FILENUMBER>2</FILENUMBER>
                <FILENAME>Sinatra_2.wav</FILENAME>
        </FILE>
        <FILE type=«other»>
                <FILENUMBER>3</FILENUMBER>
                <FILENAME>Sinatra_3.wav</FILENAME>
        </FILE>
        <ID>73365869</ID>
</LINK>
```

3.3 Fragment <link> de «Sinatra_3.wav»

```
<LINK>
        <FILE type=«other»>
                <FILENUMBER>1</FILENUMBER>
                <FILENAME>Sinatra_1.wav</FILENAME>
        </FILE>
        <FILE type=«other»>
                <FILENUMBER>2</FILENUMBER>
                <FILENAME>Sinatra_2.wav</FILENAME>
        </FILE>
        <FILE type=«actual»>
                <FILENUMBER>3</FILENUMBER>
                <FILENAME>Sinatra_3.wav</FILENAME>
        </FILE>
        <ID>73365869</ID>
<LINK>
```

Pièce jointe 6
de l'Annexe 1
(Normative)

Conventions des noms de fichier

1 Généralités

L'échange général de fichiers audio suppose que les fichiers doivent être lisibles sur ordinateur ou sur des types de système d'exploitation qui peuvent être très différents du système d'origine. Si le nom de fichier n'est pas approprié, il se peut que le fichier ne puisse pas être reconnu par le système de destination. Par exemple, certains systèmes d'exploitation limitent le nombre de caractères dans un nom de fichier. D'autres ne sont pas en mesure d'accepter des caractères multi-octets. Certains caractères ont une signification spéciale dans certains systèmes d'exploitation et devraient donc être évités. Ces lignes directrices visent à identifier les meilleures pratiques pour permettre la généralisation des échanges internationaux.

2 Longueur du nom de fichier

Les noms de fichier BWF ne devraient pas comporter plus de 31 caractères, y compris l'extension du nom de fichier.

3 Extension du nom de fichier

Les fichiers BWF doivent utiliser la même extension de nom de fichier à 4 caractères «.wav», comme un fichier WAVE traditionnel. Cela permet de pouvoir lire le contenu audio sur la plupart des ordinateurs, sans logiciel supplémentaire. Les réalisations pratiques devraient également accepter d'autres extensions, comme «.bwf», qui peuvent avoir été utilisées par erreur.

4 Ensemble de caractères nom de fichier

Les noms de fichier pour l'échange international devraient utiliser uniquement des caractères ASCII (ISO/CEI 646) de 7 bits compris entre 32 et 126 (valeur décimale).

Caractère	Valeur décimale	Valeur hexadécimale
(Espace)	32	0x20
…	…	…
~ (tilde)	126	0x7E

De plus, les caractères ci-après sont réservés pour des fonctions spéciales dans certains systèmes de fichier et ne devraient pas être utilisés dans les noms de fichier:

Caractère	Valeur décimale	Valeur hexadécimale
"	34	0x22
*	42	0x2A
/	47	0x2F
:	58	0x3A
<	60	0x3C
>	62	0x3E
?	63	0x3F
\	92	0x5C
|	124	0x7C

De plus, les caractères ci-après ne devraient pas être utilisés comme premier ou dernier caractère d'un nom de fichier:

Caractère	Valeur décimale	Valeur hexadécimale
(Espace)	32	0x20
(Point)	46	0x2E

Annexe 2

Spécification du format d'onde de radiodiffusion pour signaux audio à codage MPEG-1

Format des fichiers de données audio en radiodiffusion

1 Introduction

La présente Annexe contient la spécification relative à l'utilisation du format BWF pour l'acheminement des signaux audio à codage MPEG seulement. Pour ces signaux, il est nécessaire d'ajouter les informations suivantes aux fragments de base qui sont spécifiés dans le corps de cette Recommandation:

– une extension du fragment de format;
– un fragment factuel;
– un fragment d'extension MPEG.

L'extension du fragment de format et le fragment factuel sont l'un et l'autre spécifiés dans le cadre du format WAVE. Les informations correspondantes sont données dans la Pièce jointe 1 de l'Annexe 2.

La spécification du fragment d'extension MPEG est donnée dans le § 2 de l'Annexe 2.

Le corps de la présente Recommandation contient la spécification du fragment d'extension de données audio de radiodiffusion qui est utilisé dans tous les fichiers au format BWF. Les informations relatives au format RIFF de base sont dans la Pièce jointe 1 de la présente Annexe 2.

2 Audio MPEG

Microsoft® a spécifié la façon dont les données audio MPEG peuvent être organisées en fichiers WAVE. Une extension du fragment de format et un fragment factuel acheminent les informations nécessaires pour spécifier les options de codage MPEG. Les principes généraux sont donnés dans la Pièce jointe 1 de l'Annexe 1 et les détails dans la Pièce jointe 1 de l'Annexe 2. Pour la couche II du format MPEG, on a constaté qu'il fallait acheminer des informations supplémentaires concernant le codage du signal. Ces informations sont insérées dans le fragment d'extension de données audio MPEG <**MPEG Audio Extension**>, mis au point par le groupe d'intérêts pour les données audio de couche 2 MPEG. Ce fragment est spécifié ci-dessous.

2.1 Fragment d'extension de données audio MPEG

Le fragment d'extension de données audio MPEG est défini comme suit:

```
typedef struct {
     DWORD  ckID;              /* (mpeg_extension)ckID='mext' */
     DWORD  ckSize;            /* longueur du fragment d'extension:
                                  cksize =000C*/
     BYTE   ckData[ckSize];    /*   données du fragment */
}

typedef struct mpeg_audio_extension {
WORD SoundInformation;       /* informations supplémentaires sur le son */
WORD FrameSize;              /* longueur nominale d'une trame */
WORD AncillaryDataLength;    /* longueur des données auxiliaires */
WORD AncillaryDataDef;       /* type de données auxiliaires */
CHAR Reserved [4];           /* «NULL» */
} MPEG_EXT ;
```

Champ	Description
SoundInformation	Champ de 16 bits donnant des informations supplémentaires sur le fichier audio: Pour la couche II (ou la couche I) MPEG: Bit 0: ‘1’ Données audio homogènes ‘0’ Données audio non homogènes Les bits 1 et 2 sont utilisés pour insérer des informations additionnelles pour fichiers audio homogènes: Bit 1: ‘0’ Un bit de bourrage est utilisé dans le fichier et peut donc alterner entre ‘0’ ou ‘1’

‘1’ Le bit de bourrage est mis à ‘0’ dans tout le fichier

Bit 2: ‘1’ Le fichier contient une séquence de trames dont le bit de bourrage est mis à ‘0’ et dont la fréquence d'échantillonnage est égale à 22,05 ou 44,1 kHz.

NOTE – Un tel fichier n'est pas conforme à la norme MPEG (§ 2.4.2.3, définition du bit de bourrage) mais peut être considéré comme un cas particulier du débit binaire variable. Il n'est pas nécessaire d'avoir un décodeur MPEG pour décoder correctement un tel flux binaire, car la plupart des décodeurs rempliront cette fonction. Le débit sera légèrement inférieur à la valeur indiquée dans l'en-tête.

Bit 3: ‘1’ Utilisation d'un format libre

‘0’ Aucune trame audio à format libre

FrameSize — Nombre d'octets d'une trame nominale, codé sur 16 bits.
Ce champ n'a de sens que pour des fichiers homogènes; sinon, il est forcé à ‘0’.

Si le bit de bourrage n'est pas utilisé, c'est-à-dire si ce bit reste constant dans toutes les frames du fichier audio, le champ <FrameSize> contient la même valeur que le champ <nBlockAlign> du fragment de format. Si le bit de bourrage est utilisé et que les données audio se présentent en longueurs variables, le champ <FrameSize> indique la longueur d'une trame dont le bit de bourrage est mis à ‘0’. La longueur d'une trame dont le bit de bourrage est mis à ‘1’ est d'un octet supplémentaire (de quatre octets supplémentaires pour la couche I), c'est-à-dire <FrameSize+1>.

Le fait que le champ <nBlockAlign> soit mis à ‘1’ indique que les trames ont des longueurs variables (FrameSize ou FrameSize+1) avec un bit de bourrage variable.

AncillaryDataLength — Nombre codé sur 16 bits, indiquant le nombre minimal d'octets connus pour les données auxiliaires contenues dans le fichier audio complet. Cette valeur est calculée à partir de la fin de la trame audio.

AncillaryDataDef — Cette valeur de 16 bits spécifie comme suit le contenu des données auxiliaires:

Bit 0 mis à ‘1’: L'énergie du canal gauche est présente dans les données auxiliaires

Bit 1 mis à ‘1’: Octet privé, libre pour usage interne dans les données auxiliaires

Bit 2 mis à ‘1’: L'énergie du canal droit est présente dans les données auxiliaires

Bit 3 mis à ‘0’: Champ réservé pour usage futur de données ADR

Bit 4 mis à ‘0’: Champ réservé pour usage futur de données DAB

Bit 5 mis à ‘0’: Champ réservé pour usage futur de données J52

Bits 6 à 15 mis à ‘0’: Champs réservés pour usage futur

NOTES

– Les éléments présents dans les données auxiliaires suivent le même ordre que les positions binaires du champ, AncillaryDataDef. Le premier élément est enregistré à la fin des données auxiliaires, le deuxième juste avant le premier, etc., d'arrière en avant.

– Pour un fichier mono, le bit 2 est toujours mis à '0' et le bit 0 concerne l'énergie de la trame monophonique.

– Pour un fichier stéréo, si le bit 2 est mis à '0' et le bit 0 à 1, l'énergie concerne la valeur maximale des canaux gauche et droit.

– L'énergie est enregistrée dans 2 octets et correspond à la valeur absolue de l'échantillon de longueur maximale utilisé pour coder la trame. Il s'agit d'une valeur codée sur 15 bits en format «gros-boutiste» (Big Endian) (premier octet supérieur au second).

Reserved — Champ de 4 octets réservé pour usage futur. Ces 4 octets doivent être forcés à néant. Dans tout usage futur, la valeur nulle sera utilisée comme valeur par défaut afin d'assurer la compatibilité.

Pièce jointe 1 de l'Annexe 2 (Informative)

Format RIFF de fichier audio (.WAV)

La présente Pièce jointe spécifie les informations supplémentaires qui sont nécessaires à un fichier WAVE contenant des données audio MPEG.

Les informations contenues dans cette Pièce jointe sont extraites des spécifications du format de fichier RIFF de Microsoft®. Elles ne figurent ici qu'à titre d'information.

1 Format des fichiers (audio seulement) MPEG-1

1.1 Fragment factuel

Ce fragment est nécessaire pour tous les formats WAVE autres que WAVE_FORMAT_PCM. On y enregistre des informations variables selon les fichiers au sujet du contenu des données WAVE. Il spécifie actuellement la durée des données en nombre d'échantillons.

NOTE – Voir également la Pièce jointe 1 de l'Annexe 1, § 2.5.

1.2 En-tête de format WAVE

```
#define WAVE_FORMAT_MPEG  (0x0050)

typedef struct mpeg1waveformat_tag {
     WAVEFORMATEX            wfx;
     WORD    fwHeadLayer;
     DWORD   dwHeadBitrate;
     WORD    fwHeadMode;
     WORD    fwHeadModeExt;
     WORD    wHeadEmphasis;
     WORD    fwHeadFlags;
     DWORD   dwPTSLow;
     DWORD   dwPTSHigh;
} MPEG1WAVEFORMAT;
```

Champ	Description
wFormatTag	Ce champ doit être mis à la valeur WAVE_FORMAT_MPEG *[0x0050]*.
nChannels	Nombre de voies représentées dans les données audio: 1 pour la monophonie et 2 pour la stéréophonie.
nSamplesPerSec	Fréquence d'échantillonnage (Hz) du fichier audio: 3 000, 44 100 ou 48 000, etc. On notera cependant que si la fréquence d'échantillonnage des données est variable, ce champ doit être mis à zéro. Il est fortement recommandé d'utiliser une fréquence d'échantillonnage constante pour les applications de bureau.
nAvgBytesPerSec	Débit moyen des données; si le codage à débit binaire variable est utilisé en couche III, la valeur de ce champ peut ne pas désigner un débit conforme MPEG.
nBlockAlign	Alignement de bloc (octets) des données audio contenues dans le fragment <**data-ck**>. Pour les flux audio qui ont une longueur fixe de trame audio, l'alignement de bloc est égal à la longueur de la trame. Pour les flux dans lesquels la longueur des trames varie, le champ <**nBlockAlign**> doit être mis à '1'. Avec une fréquence d'échantillonnage de 32 ou de 48 kHz, la longueur d'une trame audio MPEG est fonction du débit binaire. Si un flux audio utilise un débit constant, la longueur des trames audio est constante. Les formules suivantes s'appliquent donc: Couche I: nBlockAlign = 4*(int)(12*BitRate/SamplingFreq) Couches II et III: nBlockAlign = (int)(144*BitRate/SamplingFreq) Exemple 1: pour la couche I, avec une fréquence d'échantillonnage de 32 000 Hz et un débit de 256 kbit/s, nBlockAlign = 384 octets. Si un flux audio contient des trames à différents débits binaires, la longueur de ces trames varie à l'intérieur de ce flux. Des longueurs de trame variables se produisent également lorsque l'on utilise une fréquence d'échantillonnage de 44,1 kHz: pour conserver la valeur nominale de ce débit, la longueur de chaque trame audio MPEG est périodiquement augmentée d'un «cran» (4 octets dans la couche I, 1 octet dans les couches II et III) par rapport aux formules indiquées ci-dessus. Dans ces deux cas, la notion d'alignement de bloc est invalide. La valeur du champ **<nBlockAlign>** doit donc être mise à 1, de manière que les applications compatibles MPEG puissent préciser si les données sont alignées en bloc ou non.

NOTE – Il est possible de construire un flux audio possédant des trames audio de longueur constante à 44,1 kHz, en donnant au bit de bourrage contenu dans chaque en-tête de trame audio la même valeur (soit 0 soit 1). Cependant, le débit du flux résultant ne correspondra pas exactement à la valeur nominale indiquée dans l'en-tête de trame et certains décodeurs n'auront peut-être pas la capacité de décoder correctement le flux. Aux fins de la normalisation et de la compatibilité, cette méthode est déconseillée.

WBitsPerSample	Champ non utilisé, mis à zéro.
CbSize	Longueur (en octets) de l'information étendue selon la structure WAVEFORMATEX. Dans le format normalisé WAVE_FORMAT_-MPEG, cette longueur est de 22 octets (0x0016). Si des champs supplémentaires sont ajoutés, cette valeur augmente.

fwHeadLayer — Couche de données audio MPEG, définie par les fanions suivants:

ACM_MPEG_LAYER1 – couche I.

ACM_MPEG_LAYER2 – couche II.

ACM_MPEG_LAYER3 – couche III

Certains flux conformes MPEG peuvent contenir des trames de différentes couches. Dans ce cas, les fanions ci-dessus doivent être mis à OR ensemble, de manière qu'un pilote puisse déterminer quelles sont les couches présentes dans le flux.

dwHeadBitrate — Débit des données, en bits par seconde. Cette valeur doit correspondre à un débit normalisé selon la spécification MPEG; tous les débits ne sont pas valides pour tous les modes et toutes les couches. Voir les Tableaux 1 et 2. On notera que ce champ enregistre le débit binaire réel et non pas le code contenu dans l'en-tête de trame MPEG. Si le débit est variable, ou s'il s'agit d'un débit non normalisé, ce champ doit être mis à zéro. Il est recommandé d'éviter, chaque fois que possible, le débit variable.

fwHeadMode — Mode de flux, défini par les fanions suivants:

ACM_MPEG_STEREO – stéréophonie.

ACM_MPEG_JOINTSTEREO – stéréophonie mixte (à réduction des sons communs, selon la puissance ou par diaphonie dynamique).

ACM_MPEG_DUALCHANNEL – son bicanal (par exemple, un flux bilingue).

ACM_MPEG_SINGLECHANNEL – son monocanal.

Certains flux conformes MPEG peuvent contenir des trames de différents modes. Dans ce cas, les fanions ci-dessus doivent être mis à OR ensemble, de façon qu'un pilote puisse préciser quels modes sont présents dans le flux. Cette situation est particulièrement probable en codage stéréophonique mixte car les codeurs peuvent juger utile de commuter dynamiquement entre le mode stéréophonique pur et le mode stéréophonique mixte en fonction des caractéristiques du signal. Dans ce cas, les deux fanions ACM_MPEG_STEREO et ACM_MPEG_JOINTSTEREO doivent être activés.

fwHeadModeExt — Ce champ contient des paramètres supplémentaires pour le codage en mode stéréophonique mixte; il n'est pas utilisé pour les autres modes. Voir le Tableau 3. Certains flux conformes MPEG peuvent contenir des trames ayant différentes extensions de mode. Dans ce cas, les valeurs indiquées dans le Tableau 3 peuvent être mises à OR ensemble. On notera que le champ fwHeadModeExt n'est utilisé que pour le codage en mode stéréophonique mixte; il doit être mis à zéro pour les autres modes (monocanal, bicanal, stéréophonie).

En général, les codeurs commutent dynamiquement entre les diverses valeurs d'extension de mode possibles, en fonction des caractéristiques du signal. Pour le codage stéréophonique mixte normal, ce champ doit donc être mis à 0x000F. Si toutefois il est souhaitable de limiter le codeur à un type particulier de codage stéréophonique mixte, ce champ peut être utilisé pour spécifier les types admissibles.

wHeadEmphasis — Ce champ décrit la désaccentuation requise par le décodeur; cela implique que l'accentuation soit appliquée au flux avant son codage. Voir le Tableau 4.

fwHeadFlags — Ce champ active les fanions suivants dans l'en-tête de trame audio:

ACM_MPEG_PRIVATEBIT – active le bit privé.

ACM_MPEG_COPYRIGHT – active le bit de copyright.

ACM_MPEG_ORIGINALHOME – active le bit d'origine.

ACM_MPEG_PROTECTIONBIT – active le bit de protection et insère dans chaque trame un code de protection contre les erreurs de 16 bits.

ACM_MPEG_ID_MPEG1 – met à 1 le bit d'identificateur qui définit le flux comme étant de type audio MPEG-1. *Ce fanion doit toujours être activé explicitement afin d'assurer la compatibilité avec les futures extensions audio MPEG (c'est-à-dire avec MPEG-2).*

Un codeur utilisera la valeur de ces fanions pour activer les bits correspondants dans l'en-tête de chaque trame audio MPEG. Lorsqu'ils décrivent un flux de données codées, ces fanions représentent un OR logique entre les fanions activés dans chaque en-tête de trame. En d'autres termes, si le bit de copyright est activé dans un ou plusieurs en-têtes de trame du flux, le fanion ACM_MPEG_COPYRIGHT sera activé. La valeur de ces fanions n'est donc pas forcément valide pour chaque trame audio.

dwPTSLow — Ce champ (de même que le champ suivant) se compose du pointeur temporel de présentation (PTS) de la première trame du flux audio, extraite de la couche système MPEG. Le champ dwPSTLow contient les 32 bits de poids faible du pointeur PTS de 33 bits. Le pointeur PTS peut être utilisé pour faciliter la réintégration d'un flux audio dans un flux vidéo associé. Si le flux audio n'est pas associé à une couche système, ce champ doit être mis à zéro.

dwPTSHigh — Ce champ (de même que le champ précédent) se compose du pointeur temporel de présentation (PTS) de la première trame du flux audio, extraite de la couche système MPEG. La position binaire de poids faible du champ dwPSTHigh contient le bit de poids fort du pointeur PTS de 33 bits. Le pointeur PTS peut être utilisé pour faciliter la réintégration d'un flux audio dans un flux vidéo associé. Si le flux audio n'est pas associé à une couche système, ce champ doit être mis à zéro.

NOTE – Les deux champs précédents peuvent être traités comme un seul entier codé sur 64 bits; facultativement, le champ dwPTSHigh peut être testé en tant que fanion pour déterminer si le bit de poids fort est activé ou non.

TABLEAU 1

Débits admissibles (bit/s)

Code d'en-tête de trame MPEG	Couche I	Couche II	Couche III
‘0000’	format libre	format libre	format libre
‘0001’	32 000	32 000	32 000
‘0010’	64 000	48 000	40 000
‘0011’	96 000	56 000	48 000
‘0100’	128 000	64 000	56 000
‘0101	160 000	80 000	64 000
‘0110’	192 000	96 000	80 000
‘0111’	224 000	112 000	96 000
‘1000’	256 000	128 000	112 000
‘1001’	288 000	160 000	128 000
‘1010’	320 000	192 000	160 000
‘1011’	352 000	224 000	192 000
‘1100’	384 000	256 000	224 000
‘1101’	416 000	320 000	256 000
‘1110’	448 000	384 000	320 000
‘1111’	interdit	interdit	interdit

TABLEAU 2

Combinaisons de débits et de modes admissibles pour la Couche II

Débit (bit/s)	Modes admissibles
32 000	monocanal
48 000	monocanal
56 000	monocanal
64 000	tous modes
80 000	monocanal
96 000	tous modes
112 000	tous modes
128 000	tous modes
160 000	tous modes
192 000	tous modes
224 000	stéréo, stéréo mixte (puissance), bicanal
256 000	stéréo, stéréo mixte (puissance), bicanal
320 000	stéréo, stéréo mixte (puissance), bicanal
384 000	stéréo, stéréo mixte (puissance), bicanal

TABLEAU 3

Extension de mode

fwHeadModeExt	Code d'en-tête de trame MPEG	Couches I et II	Couches III
0x0001	‘00’	sous-bandes 4 à 31 en stéréo mixte (en puissance)	pas de codage de stéréo mixte (en puissance) ni de stéréo à signaux *M* et *S* multiplexés
0x0002	‘01’	sous-bandes 8 à 31 en stéréo mixte (puissance)	stéréo mixte (puissance)
0x0004	‘10’	sous-bandes 12 à 31 en stéréo mixte (puissance)	stéréo M/S
0x0008	‘11’	sous-bandes 16 à 31 en stéréo mixte (puissance)	codage de stéréo mixte (puissance) et de stéréo M/S

TABLEAU 4

Champ d'accentuation

wHeadEmphasis	Code d'en-tête de trame MPEG	Désaccentuation requise
1	‘00’	pas d'accentuation
2	‘01’	accentuation de 50/15 μs
3	‘10’	champ réservé
4	‘11’	Rec. UIT-T J.17

1.3 Fanions utilisés dans les champs de données

fwHeadLayer

Les fanions suivants sont définis pour le champ <fwHeadLayer>. Pour le codage, l'un de ces fanions doit être activé, de façon à indiquer au codeur la couche à utiliser. Pour le décodage, le pilote peut vérifier ces fanions afin de déterminer s'il est en mesure de décoder le flux. On notera qu'un flux conforme MPEG peut utiliser différentes couches dans différentes trames d'un même flux. Plusieurs de ces fanions peuvent donc être activés en même temps.

```
#define ACM_MPEG_LAYER1          (0x0001)
#define ACM_MPEG_LAYER2          (0x0002)
#define ACM_MPEG_LAYER3          (0x0004)
```

fwHeadMode

Les fanions suivants sont définis pour le champ <fwHeadMode>. Pour le codage, l'un de ces fanions doit être activé, de façon à indiquer au codeur le mode à utiliser; pour le codage stéréophonique mixte, les deux fanions ACM_MPEG_STEREO et ACM_MPEG_JOINTSTEREO seront normalement activés, de manière que le codeur puisse n'utiliser le codage stéréophonique mixte que si celui-ci est plus efficace que le codage stéréophonique classique. Pour le décodage, le pilote peut vérifier ces fanions afin de déterminer s'il est en mesure de décoder le flux. On notera qu'un flux conforme MPEG peut utiliser différentes couches dans différentes trames d'un même flux. Plusieurs de ces fanions peuvent donc être activés en même temps.

```
#define ACM_MPEG_STEREO          (0x0001)
#define ACM_MPEG_JOINTSTEREO   (0x0002)
#define ACM_MPEG_DUALCHANNEL   (0x0004)
#define ACM_MPEG_SINGLECHANNEL (0x0008)
```

fwHeadModeExt

Le Tableau 3 définit les fanions pour le champ <fwHeadModeExt>. Ce champ n'est utilisé que pour le codage stéréophonique mixte; pour les autres modes de codage, ce champ doit être mis à zéro. Pour le codage stéréophonique mixte, ces fanions indiquent les types de codage stéréophonique mixte qu'un codeur est autorisé à utiliser. Normalement, un codeur sélectionnera dynamiquement l'extension de mode la mieux appropriée au signal d'entrée; normalement, une application mettra donc ce champ à la valeur 0x000f, de façon que le codeur puisse sélectionner entre toutes les possibilités. Il est toutefois possible de limiter le codeur en éliminant certains de ces fanions. Pour un flux codé, ce champ indique les valeurs du champ MPEG *mode_extension* qui sont présentes dans le flux.

fwHeadFlags

Les fanions suivants sont définis pour le champ <fwHeadFlags>. Ces fanions doivent être activés avant le codage, de façon que les bits appropriés soient activés dans l'en-tête de trame MPEG. Lors de la description d'un flux audio codé MPEG, ces fanions représentent un OR logique entre les bits correspondants de l'en-tête de chaque trame audio. En d'autres termes, si le bit est activé dans une des trames, il l'est également dans le champ <fwHeadFlags>. Si une application enroule un en-tête de fichier WAVE au format RIFF autour d'un flux de données audio binaires précodé MPEG, cette application est chargée d'analyser les éléments de ce flux binaire et d'activer les fanions dans ce champ.

```
#define ACM_MPEG_PRIVATEBIT      (0x0001)
#define ACM_MPEG_COPYRIGHT       (0x0002)
#define ACM_MPEG_ORIGINALHOME    (0x0004)
#define ACM_MPEG_PROTECTIONBIT (0x0008)
#define ACM_MPEG_ID_MPEG1        (0x0010)
```

1.4 Données audio dans les fichiers MPEG

Le <fragment de données> (<**data chunk**>) se compose d'une séquence de données audio MPEG-1 telle que définie dans l'ISO 11172, Partie 3 (Audio). Cette séquence comporte un flux binaire qui est enregistré dans le fragment de données sous la forme d'un ensemble d'octets. À l'intérieur d'un de ces octets, le bit de poids fort est le premier du flux et le bit de poids faible le dernier. Les données ne sont *pas* inversées dans les octets. Par exemple, les données suivantes sont codées sur les 16 premiers bits (de gauche à droite) d'un en-tête de trame audio normal:

Syncword	ID	Layer	ProtectionBit	…
111111111111	1	10	1	…

Ces données seront enregistrées dans des octets selon l'ordre suivant:

Byte0	Byte1	...
FF	FD	...

1.4.1 Trames audio MPEG

Une séquence de données audio MPEG se compose d'une série de trames audio dont chacune commence par un en-tête de trame. La plupart des champs contenus dans cet en-tête de trame correspondent à des champs dans la structure MPEG1WAVEFORMAT définie ci-dessus. Pour le codage, ces champs peuvent être activés dans la structure MPEG1WAVEFORMAT et le pilote peut utiliser ces informations pour activer les bits appropriés dans l'en-tête de trame au moment du codage. Pour le décodage, le pilote peut vérifier ces champs afin de déterminer s'il est en mesure de les décoder.

1.4.2 Codage

Un pilote qui code un flux audio MPEG doit lire les champs d'en-tête dans la structure MPEG1WAVEFORMAT puis activer les bits correspondants dans l'en-tête de trame MPEG. Si un pilote a besoin de quelques autres informations, il doit les obtenir soit à partir d'une fenêtre de dialogue de configuration ou au moyen d'une fonction de reprise du pilote. On trouvera de plus amples informations dans le paragraphe ci-après, consacré aux données auxiliaires.

Si un flux audio à précodage MPEG est multiplexé avec un en-tête de format RIFF, il appartient à l'application d'analyser les éléments constituants du flux binaire et d'activer les champs correspondants dans la structure MPEG1WAVEFORMAT. Si la fréquence d'échantillonnage ou la valeur de débit binaire n'est pas constante dans tout le flux de données, le pilote doit mettre à zéro les champs MPEG1WAVEFORMAT correspondants (<nSamplesPerSec> et <dwHeadBitrate>), comme décrit ci-dessus. Si le flux contient des trames appartenant à plusieurs couches, il doit activer les fanions du champ <fwHeadLayer> pour toutes les couches présentes dans le flux. Étant donné que des champs comme <fwHeadFlags> peuvent varier d'une trame à l'autre, il faut prendre des précautions lors de l'activation et du contrôle de ces fanions; en général, une application ne doit pas compter qu'ils seront valides pour chaque trame. Lors de l'activation de ces fanions, il convient d'observer les lignes directrices suivantes:

– Le fanion ACM_MPEG_COPYRIGHT doit être activé si l'une des trames du flux contient le bit de copyright activé.

– Le fanion ACM_MPEG_PROTECTIONBIT doit être activé si l'une des trames du flux contient le bit de protection activé.

– Le fanion ACM_MPEG_ORIGINALHOME doit être activé si l'une des trames du flux contient le bit d'origine activé. Ce bit peut être annulé si une copie du flux est faite.

– Le fanion ACM_MPEG_PRIVATEBIT doit être activé si l'une des trames du flux contient le bit privé activé.

– Le fanion ACM_MPEG_ID_MPEG1 doit être activé si l'une des trames du flux contient le bit d'identification activé. Pour les flux MPEG-1 streams, le bit d'identification doit toujours être activé; de futures extensions du format MPEG (comme le format multicanaux MPEG-2) pourront toutefois avoir le bit d'identification annulé.

Si le flux audio MPEG est extrait d'un flux MPEG de couche système, ou si le flux est destiné à être intégré dans la couche système, les champs de PTS peuvent être utilisés. Le pointeur PTS est un champ de la couche système MPEG qui est utilisé pour la synchronisation des divers champs. Il est codé sur 33 bits et l'en-tête du format RIFF des fichiers WAVE enregistre donc sa valeur en deux fois: le champ <dwPTSLow> contient les 32 bits de poids faible du pointeur PTS et le champ <dwPTSHigh> contient le bit de poids fort. Ces deux champs peuvent être traités ensemble comme un entier de 64 bits; en option, le champ <dwPTSHigh> peut être contrôlé en tant que fanion pour déterminer si le bit de poids fort est activé ou annulé. Lors de l'extraction d'un flux audio d'une couche système, un pilote doit activer les champs de pointeur PTS en fonction des pointeurs PTS de la première trame des données audio. Cette information pourra être utilisée ultérieurement pour

réintégrer le flux audio dans la couche système. *Les champs PTS ne doivent pas être utilisés à d'autres fins*. Si le flux audio n'est pas associé à la couche système MPEG, les champs PTS doivent être mis à zéro.

1.4.3 Décodage

Un pilote peut contrôler les champs contenus dans la structure MPEG1WAVEFORMAT afin de déterminer s'il est en mesure de décoder le flux. Il doit cependant tenir compte du fait que certains champs, comme le champ <fwHeadFlags>, peuvent ne pas être constants d'une trame à l'autre du flux binaire. Un pilote ne doit jamais utiliser les champs de la structure MPEG1WAVEFORMAT pour effectuer le décodage proprement dit. Les paramètres de décodage doivent être entièrement extraits du flux de données MPEG.

Un pilote peut contrôler le champ <nSamplesPerSec> afin de déterminer s'il est compatible avec la fréquence d'échantillonnage spécifiée. Si le flux MPEG contient des données à fréquence d'échantillonnage variable, le champ <nSamplesPerSec> doit être mis à zéro. Si le pilote ne peut pas traiter ce type de flux de données, il ne doit pas essayer de les décoder mais doit immédiatement arrêter de fonctionner.

1.5 Données auxiliaires

Les données audio contenues dans une trame audio MPEG ne remplissent pas toujours la totalité de la trame. Les données restantes sont dites *auxiliaires*: elles peuvent avoir tout format désiré et peuvent être utilisées pour acheminer des informations supplémentaires d'un type quelconque. Si un pilote souhaite prendre en charge les données auxiliaires, il doit avoir la capacité d'acheminer ces données en provenance et à destination de l'application appelante. Le pilote peut, à cette fin, faire appel à une fonction de reprise. En principe, le pilote peut appeler une fonction de reprise spécifiée chaque fois qu'il est en possession de données auxiliaires à transmettre à l'application (c'est-à-dire à décoder) ou chaque fois qu'il a besoin d'autres données auxiliaires (à coder).

Les pilotes doivent tenir compte du fait que toutes les applications ne seront pas disposées à traiter les données auxiliaires. Un pilote ne doit donc fournir ce service que lorsque l'application lui en fait la demande expresse. Le pilote peut définir un message particulier qui active et désactive la fonction de reprise. Des messages distincts peuvent être définis pour les opérations de codage et de décodage, afin d'augmenter la flexibilité.

On notera que cette méthode n'est pas forcément appropriée à tous les pilotes ou à toutes les applications; elle n'est présentée qu'à titre d'illustration de la façon dont des données auxiliaires peuvent être prises en charge.

NOTE – On trouvera de plus amples informations sur les données auxiliaires dans le fragment <**MPEG_Audio_Extension chunk**> qu'il y a lieu d'utiliser pour les fichiers MPEG conformes au format BWF. Voir le § 2 de l'Annexe 2.

Références

ISO/CEI 11173-3: MPEG 1.

ISO/CEI 13818-3: MPEG 2.

NOTE – Les documents Microsoft® sont disponibles à l'adresse Internet suivante: http://www.microsoft.com.

Annexe 3

Spécification du format BWF

Format de fichier de données audio pour la radiodiffusion

SPÉCIFICATIONS DES MÉTADONNÉES

1 Introduction

La présente Annexe spécifie l'utilisation du format BWF pour l'acheminement d'informations sur des éléments audio traités sur une station de travail audionumérique (Fig. 2). Le fichier BWF est utilisé comme conteneur indépendant de la plate-forme pour le signal sonore et toutes les métadonnées associées. Le serveur où sont archivés les éléments reçus peut extraire les informations requises du fichier correspondant et les utiliser en fonction des besoins (exemple: transfert dans une base de données, etc.) (Fig. 3).

FIGURE 2

Transfert sur fichier BWF des signaux saisis sur le poste de travail

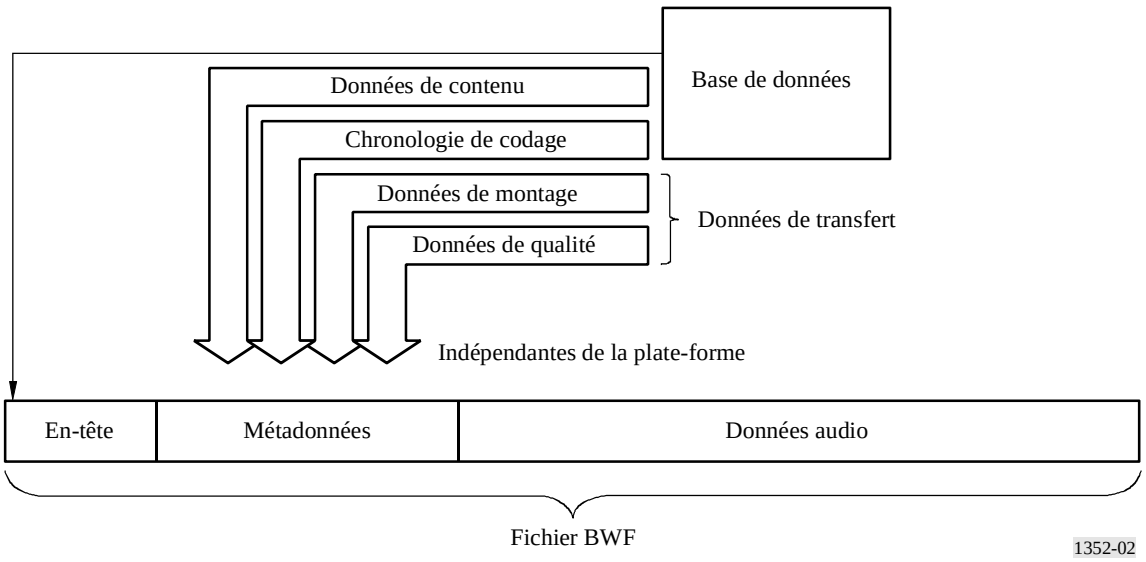
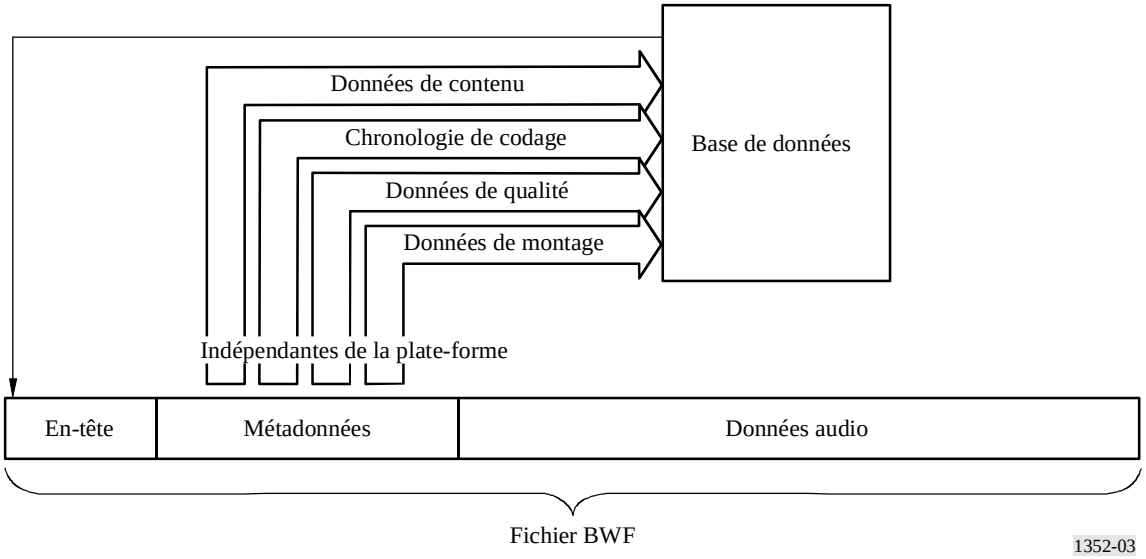


FIGURE 3

Extraction de données d'un fichier BWF par le serveur où sont archivés les programmes reçus



La présente Annexe spécifie un nouveau fragment servant à acheminer des informations qui ne sont pas déjà présentes dans un fichier BWF de base. Elle vise aussi à expliquer comment les fragments présents dans le fichier BWF doivent être utilisés.

Certaines précautions sont à prendre lors du montage d'un programme sur fichier BWF contenant des rapports de qualité. Lorsque plusieurs fichiers BWF sont combinés au niveau du système de montage, la liste de décision d'édition (EDL) doit désigner les éléments appropriés des fragments Coding History et Quality de chaque fichier source BWF. Par ailleurs, lorsqu'un nouveau fichier est établi à partir d'éléments de plusieurs fichiers, il faut associer à ce nouveau fichier de nouveaux fragments Coding History et Quality.

2 Rapport de saisie

Pour sauvegarder les archives analogiques ou numériques d'origine, il importe de réenregistrer le signal sonore d'origine dans le fichier BWF, au niveau de qualité intégral. Le rapport de saisie contient des informations sur l'ensemble de la chaîne de traitement, du domaine analogique au domaine numérique, ou sur les transferts effectués dans le domaine numérique (par exemple, à partir d'un support disque compact (CD) ou d'une bande magnétique audionumérique (DAT).

Le rapport de saisie et les données d'analyse du signal audio font partie des métadonnées du fichier BWF.

Le rapport de saisie comprend trois parties:

– le champ CodingHistory du fragment <bext> du fichier BWF. Ce champ contient des données détaillées sur l'ensemble de la chaîne de transmission telles que type de bande magnétique, type de support – disque compact ou cassette DAT – fichier BWF (chronologie du signal sonore);

– le rapport Quality Report figurant dans le fragment <qlty>. Ce rapport contient des informations décrivant la totalité des événements relevant de la qualité du signal sonore enregistré dans le fragment de données «wave». Chaque événement reconnu par l'opérateur ou par l'ordinateur figure dans ce rapport, avec tous les détails utiles: type d'événement, indications horaires exactes, priorité, situation. Les paramètres de qualité globale etc. sont également indiqués;

– l'élément «Cue Sheet» du fragment <qlty> est une liste d'événements qui comporte les indications horaires exactes et une description complète du signal sonore (début d'un aria, point de départ d'un discours important etc.). Les archivistes peuvent ainsi compléter les métadonnées de la base de données à l'aide d'outils informatisés.

2.1 Syntaxe du rapport de saisie

– Le rapport de saisie (*capturing report*) se compose de chaînes de caractères ASCII (ISO 646) [ISO/CEI, 1991] réparties en lignes de 256 caractères maximum.

– La commande <CR/LF> (ASCII 0Dh, 0Ah) doit figurer à la fin de chaque ligne.

– Une ligne peut contenir une ou plusieurs chaînes variables séparées par des virgules (ASCII 2Bh).

– Les chaînes variables sont en caractères ASCII et ne doivent pas contenir de virgules.

– Les éléments de séparation, dans les chaînes variables, doivent être des points-virgules (ASCII 3Bh).

3 Champ CodingHistory du fragment <bext>

Les chaînes utilisées dans le champ de chronologie de codage sont spécifiées dans la Pièce jointe 2 de l'Annexe 1. Pour plus de commodité, ces informations sont reproduites ci-après.

A=<ANALOGUE, ……..>	Information sur le trajet du signal sonore analogique
A=<PCM, ……...>	Information sur le trajet du signal sonore numérique
F=<48000, 441000, etc.>	Fréquence d'échantillonnage [Hz]
W=<16, 18, 20, 22, 24, etc.>	Longueur de mot [bits]
M=<mono, stereo, 2-channel>	Mode
T=<free ASCII-text string>	Texte pour observations

4 Quality Chunk (fragment qualité)

Le fragment «Quality Chunk» est défini dans le texte en italique du § 4.1:

4.1 Éléments du fragment «Quality Chunk»

FileSecurityReport: Ce champ contient l'élément FileSecurityCode du fragment QualityChunk. Il s'agit d'une valeur de 32 bits contenant le total de contrôle [0 ....231].

FileSecurityWave: Ce champ contient le code FileSecurityCode des données BWF Wave. Il s'agit d'une valeur de 32 bits qui contient le total de contrôle [0 ....231].

```
Quality-chunk typedef struct {
        DWORD     ckID;          /* (fragment qualité) cklD='qlty' */
        DWORD     ckSize;        /* longueur du fragment qualité */
        BYTE      ckData[ckSize]; /* données du fragment */
}

typedef struct quality_chunk {
DWORD FileSecurityReport;        /*Code FileSecurityCode du rapport de qualité */
DWORD FileSecurityWave;          /* Code FileSecurityCode des données BWF wave data */
CHAR BasicData[ ];               /* ASCII: «Données de base» */
CHAR StartModulation[] ;         /* ASCII: «Données de début de modulation» */
CHAR QualityEvent[ ];            /* ASCII: «Données d'événement de qualité» */
CHAR EndModulation[];            /* ASCII: «Données de fin de modulation» */
```

CHAR QualityParameter[ ] */* ASCII: «Données des paramètres de qualité» */*
CHAR OperatorComment[ ]; */* ASCII: «Commentaires d'opérateur» */*
CHAR CueSheet[ ]; */* ASCII: «Données de feuille de montage» */*
} quality-chunk

BasicData: **Données fondamentales de saisie.**

B= Chaîne ASCII contenant les données fondamentales du programme sonore.

Archive No. (AN): Numéro d'archive (maximum: 32 caractères).

Title (TT): Titre/numéro de prise de son des données sonores (maximum: 256 caractères).

Duration (TD): 10 caractères ASCII contenant la durée de la séquence sonore.

Format: « hh:mm:ss:d »

Heures	hh:	0…23
Minutes	mm:	0…59
Secondes	ss:	0…59
1/10 s	d:	0…9

Date (DD): 10 caractères ASCII contenant la date de numérisation.

Format: «yyyy:mm:dd»

Année	yyyy:	0000...9999
Mois	mm:	0...12
Jour	dd:	0…31

Operator (OP): Chaîne ASCII (maximum: 64 caractères) contenant le nom de la personne procédant à la numérisation.

Copying station (CS): Chaîne ASCII (maximum: 64 caractères) contenant le type et le numéro de série du poste de travail utilisé pour créer le fichier.

StartModulation: Début de modulation de l'enregistrement d'origine.

SM= 10 caractères ASCII contenant le point de départ du signal sonore à partir du démarrage du fichier

Format: «hh:mm:ss:d»

Heures	hh:	0…23
Minutes	mm:	0…59
Secondes	ss:	0…59
1/10 s	d:	0…9

Sample count (SC): Code d'adresse d'échantillon du point de début de modulation au démarrage du fichier (début de modulation en hexadécimal).

Format: «########H»

0H….. FFFFFFFFH (0….. 4.295×10^9)

Comment (T): Chaîne ASCII contenant des observations.

QualityEvent: Information décrivant chaque événement «qualité» du signal sonore. Une chaîne QualityEvent est utilisée pour chaque événement.

Q= Chaîne ASCII (maximum: 256 caractères) contenant les événements de qualité.

Event number (M): Marque numérotée apposée manuellement par l'opérateur.

Format: «M###» ###: 001…999

Event number (A): Marque numérotée générée de façon automatique par le système.

Format: «A###» ###: 001…999

Priority (PRI): Priorité de l'événement de qualité.

Format: «#» #: 1 (LO)…… 5 (HI)

Time stamp (TS): 10 caractères ASCII contenant l'indication horaire de l'événement de qualité à partir du démarrage du fichier.

Format: «hh:mm:ss:d»

Heures	hh:	0…23
Minutes	mm:	0…59
Secondes	ss:	0…59
1/10 s	d:	0…9

Event type (E): Chaîne ASCII (maximum: 16 caractères) décrivant le type d'événement, par exemple «Click» (clic), «AnalogOver» (fin de signal analogique), «Transparency» (transparence) ou les paramètres de qualité («Quality-Parameters») (définis ci-dessous) dépassant les limites fixées, par exemple «QP:Azimuth:L-20,9smp».

Status (S): Chaîne ASCII (maximum: 16 caractères) contenant l'état de traitement de l'événement, par exemple «unclear» (pas clair), «checked» (vérifié), «restored» (rétabli), «deleted» (supprimé).

Comment (T): Chaîne ASCII contenant les observations.

Sample count (SC): Code d'adresse d'échantillon d'indication horaire du démarrage du fichier (ASCII hexadécimal).

Format: «########H»

0H……FFFFFFFFH (0…… $4{,}295 \times 10^9$)

QualityParameter Paramètre de qualité décrivant le signal sonore.

P= Chaîne ASCII (maximum: 256 caractères) contenant les paramètres de qualité.

Parameters (QP):	MaxPeak (niveau crête):	–xx.x dBFSL;–yy.y dBFSR [–99,9…...–00,0]
	MeanLevel (niveau moyen):	–xx.x dBFSL;–yy.y dBFSR [–99,9…...–00,0]
	Correlation (corrélation):	±x.x [–1,0……. +1,0]
	Dynamic (dynamique):	xx.x dBL; yy.y dBR [00,0……..99,9]
	ClippedSamples (échantillons écrêtés):	xxxx smpL; yyyy smpR [0....9999]
	SNR (rapport signal/bruit):	xx.x dBL; yy.y dBR [00,0……..99,9]
	Bandwidth (largeur de bande):	xxxxx HzL; yyyyy HzR [0……....20000]
	Azimuth (azimuth):	L±xx.x smp [–99,9…..+99,9]
	Balance (balance):	L±x.x dB [–9,9……..+9,9]
	DC-Offset (décalage continu):	x.x %L; y.y %R [0,0……..9,9]

Speech (parole):	xx.x%	[0,0…….99,9]
Stereo (stéréo):	xx.x%	[0,0……..…99,9]

(L = voie gauche, R = voie droite)

Quality factor (QF): Facteur de qualité d'ensemble du fichier sonore [1 …… 5 (meilleure), 0 = indéfinie]

Inspector (IN): Chaîne ASCII (maximum: 64 caractères) contenant le nom de la personne chargée de vérifier le fichier sonore.

File status (FS): Chaîne de caractères ASCII décrivant le statut «Prêt pour transmission?».

[Y (oui) / N (non) / U: Fichier prêt/pas prêt/statut non défini]

OperatorComment: observations de l'opérateur.

T= Chaîne ASCII (maximum 256 caractères) contenant les observations.

EndModulation: fin de modulation.

EM= 10 caractères ASCII contenant l'indication de fin de modulation du signal sonore.

Format: « hh:mm:ss:d »

Heures	hh:	0…23
Minutes	mm:	0…59
Secondes	ss:	0…59
1/10 s	d:	0…9

Sample count (SC): Code d'adresse d'échantillon du point de fin de modulation (ASCII hexadécimal)

Format: « ########H »

0H……FFFFFFFFH (0……4,295 × 10^9)

Comment (T): Chaîne ASCII contenant des observations.

CueSheet: Données de feuille de montage.

C= Chaîne ASCII (maximum: 256 caractères) contenant les points de montage.

Cue number (N): Nombre de points de montage générés automatiquement par le système.

Format: «N###» ###: 001...999

Time stamp (TS): 10 caractères ASCII contenant l'indication horaire du point de montage.

Format: «hh:mm:ss:d»

Heures	hh:	0…23
Minutes	mm:	0…59
Secondes	ss:	0…59
1/10 s	d:	0…9

Text (T): Chaîne ASCII contenant les observations décrivant le point de montage, exemple: «Début d'un aria».

Sample count (SC): Code d'adresse d'échantillon du point d'indication horaire (ASCII hexadécimal)

Format: «########H»

0H…… FFFFFFFFH (0…4,295 × 10^9)

5 Exemples de rapports de saisie

5.1 Numérisation de signaux analogiques

(Informations de base contenues dans le champs CodingHistory du fragment <bext>)

Numéro de ligne

01 A=ANALOGUE, M=stereo, T=Studer A816; SN1007; 38; telcom; Agfa PER528<CR/LF>

02 A=PCM, F=48000, W=18, M=stereo, T=NVision NV 1000; A/D<CR/LF>

03 A=PCM, F=48000, W=16, M=stereo, T=nodither; DIO<CR/LF>

(Champ QualityReport du fragment qualité)

Numéro de ligne

01 <FileSecurityReport>

02 <FileSecurityWave>

03 B=CS=QUADRIGA2.0; SN10012, OP=name of operator<CR/LF>

04 B=AN=archive number, TT=title of sound<CR/LF>

05 B=DD= yyyy:mm:dd, TD=hh:mm:ss:d<CR/LF>

06 SM=00:00:04:5, T=tape noise changing to ambience, SC=34BC0H<CR/LF>

07 Q=A001, PRI=2, TS=00:01:04:0, E=Click, S=unclear, SC=2EE000H<CR/LF>

08 Q=A002, PRI=3, TS=00:12:10:3, E=DropOut, S=checked, SC=216E340H<CR/LF>

09 Q=A003, PRI=4, TS=00:14:23:0, E=Transparency, S=checked, SC=2781480H<CR/LF>

10 Q=M004, PRI=1, TS=00:18:23:1, E=PrintThrough, S=checked, SC=327EF40H<CR/LF>

11 Q=A005, PRIG, TS=00:20:01:6, E=Click0n, S=unclear, T=needs restoration, SC=3701400H<CR/LF>

12 Q=A006, PRI=5, TS=00:21:20:3, E=QP:Azimuth:L=–20.9smp, S=unclear, SC=3A9B840H<CR/LF>

13 Q=A007, PRI=3, TS=00:21:44:7, E=AnalogOver, S=checked, SC=3BB9740H<CR/LF>

14 Q=A008, TS=00:22:11:7, E=C1ickOff, SC=3BB9740H<CR/LF>

15 Q=A009, PRI=1, TS=00:28:04:0, E=DropOut, S=deleted, SC=4D16600H<CR/LF>

16 EM=00:39:01:5, T=fade-out of applause, SC=6B2F740H<CR/LF>

17 P=QP:MaxPeak:–2. 1dBFSL;–2.8dBFSR<CR/LF>

18 P=QP:MeanLevel:–11.5dBFSL; 8.3dBFSR<CR/LF>

19 P=QP:Correlation:+0.8<CR/LF>

20 P=QP:Dynamic:51.4dBL;49.6dBR<CR/LF>

21 PAP:ClippedSamples:OsmpL;OsmpR<CR/LF>

22 P=QP:SNR:32.3dBL;35.1dBR<CR/LF>

23 P=QP:Bandwidth:8687HzL;7943HzR<CR/LF>

24 P=QP:Azimuth:L–6.2smp<CR/LF>

25 P=QP:Balance L:+2.1dB<CR/LF>

26 P=QP:DC-Offset:0.0%L;0.0%R<CR/LF>

27 P=QP:Speech:64.2%<CR/LF>

28 P=QP:Stereo:89.3%<CR/LF>

29 P=QF=2<CR/LF>

30 P=IN=name of inspector<CR/LF>

31 P=FS=N<CR/LF>

(Champ CueSheet du fragment qualité)

Numéro de ligne

32 C=N001, TS=00:17:02:5, T=beginning of speech, SC=2ECE6C0 H<CR/LF>

33 C=N002, TS=00:33:19:2, T=start of aria, SC=5B84200H<CR/LF>

Interprétation de l'exemple 1

(Informations de base présentes dans le champ CodingHistory)

Ligne 1: La bande magnétique analogique Agfa PER528 est lue sur un magnétophone Studer A816, numéro de série 1007, comportant un circuit d'expansion:

Vitesse de bande: 38 cm/s
Mode: stéréo

Ligne 2: Pour la numérisation, on utilise un convertisseur analogique/numérique NVision NV1000 présentant les caractéristiques suivantes:

Fréquence d'échantillonnage: 48 kHz
Résolution de codage: 18 bits par échantillon
Mode: stéréo

Ligne 3: Le fichier d'origine est enregistré sous forme de fichier BWF linéaire, en codage MIC, sur l'entrée numérique de la station de travail de réenregistrement, sans dispersion de bruit de quantification:

Fréquence d'échantillonnage: 48 kHz
Résolution de codage: 16 bits par échantillon
Mode: stéréo

(Champ QualityReport du fragment qualité)

Lignes 1 et 2: Codes de sécurité fichier du fragment «qualité» et des données «wave».

Lignes 3 à 5: L'opérateur utilise pour le réenregistrement une station de travail QUADRIGA2.0, numéro de série 10012.

(OP). La bande comporte un numéro d'archive (AN) et un titre (TT) et a été numérisée à la date.

(DD). La durée du signal sonore dans le fichier BWF est (TD).

Ligne 6: Début de modulation (SM), indication horaire (TS), décompte d'échantillon (SC) et observations (T).

Lignes 7 à 15: Événements (E) reconnus par l'opérateur (M) et/ou la régie du système (A), avec mention de priorité (PRI) et indication horaire (TS). Le statut d'événement (S) et les observations (T) donnent des informations complémentaires. Le décompte d'échantillonnage (SC) indique l'heure précise.

Ligne 16: Fin de modulation (EM), indication horaire, décompte d'échantillon (SC) et observations (T).

Lignes 17 à 28: Paramètres de qualité du signal sonore complet, dans le fragment données «wave».

Lignes 29 à 31: La régie automatique du système indique le facteur de qualité générale (QF) et le nom de l'inspecteur (IN), et précise (FS) si la qualité du fichier sonore justifie la mention «prêt pour transmission».

(Champ CueSheet du fragment qualité)

Lignes 32 et 33: Des tops sonores indiquent le début de la parole et le début d'un aria.

5.2 Saisie sur disque audionumérique (disque compact)

(Informations de base présentes dans le champ CodingHistory du fragment <bext>)

Numéro de ligne

01 A=PCM, F=44100, W=16, M=stereo, T=SonyCDP-D500; SN2172; Mitsui CD-R74 <CR/LF>

02 A=PCM, F=48000, W=24, M=stereo, T=DCS972; D/D<CR/LF>

03 A=PCM, F=48000, W=24, M=stereo, T=nodither; DIO<CR/LF>

(Champ QualityReport du fragment qualité)

Numéro de ligne

01 <FileSecurityReport>

02 <FileSecurityWave>

etc.: Comme dans l'exemple du § 5.1 ci-dessus.

(Champ CueSheet du fragment qualité)

Comme dans l'exemple du § 5.1 ci-dessus.

Interprétation de l'exemple 2

(Informations de base présentes dans le champ CodingHistory)

Ligne 1: Un disque audionumérique (CD) enregistrable de type Mitsui CD-R74 est lu sur une platine CD Sony CDP-D500, numéro de série 2172:

Fréquence d'échantillonnage:	44,1 kHz
Résolution de codage:	16 bits par échantillon
Mode:	stéréo

Ligne 2: On utilise un convertisseur de débit d'échantillonnage DCS972 présentant les caractéristiques suivantes:

Fréquence d'échantillonnage:	48 kHz (à partir de 44,1 kHz)
Résolution de codage:	24 bits par échantillon
Mode:	stéréo

Ligne 3: Le fichier d'origine est enregistré sous forme de fichier BWF linéaire avec codage MIC sur l'entrée numérique de la station de travail de réenregistrement, sans dispersion de bruit de quantification:

Fréquence d'échantillonnage:	48 kHz
Résolution de codage:	24 bits par échantillon
Mode:	stéréo

(Champ QualityReport du fragment qualité)

Lignes 1 et 2: codes de sécurité fichier du fragment «qualité» et des données «wave».

Les autres informations sont utilisées selon le processus de saisie sur disque audionumérique comme dans l'exemple 1 du § 5.1 ci-dessus.

(Champ CueSheet du fragment qualité)

Les données de feuille de montage sont utilisées selon le processus de saisie sur disque audionumérique, comme dans l'exemple 1 du § 5.1 ci-dessus.

5.3 Saisie sur cassette DAT

(Informations de base du champ CodingHistory du fragment <bext>)

Numéro de ligne

01 A=PCM, F=48000, W=16, M=stereo, T=SonyPCM-8500; SN1037; TDK DA-R120 <CR/LF>

02 A=PCM, F=48000, W=16, M=stereo, T=no dither; DIO<CR/LF>

(Champ QualityReport du fragment qualité)

Numéro de ligne

01 <FileSecurityReport>

02 <FileSecurityWave>

etc. Comme dans l'exemple du § 5.1 ci-dessus.

(Champ CueSheet du fragment qualité)

Comme dans l'exemple du § 5.1 ci-dessus.

Interprétation de l'exemple 3

(Informations de base présentes dans le champ CodingHistory)

Ligne 1: Une cassette DAT type TDK DA-8120 est lue sur un enregistreur DAT Sony PCM-8500, numéro de série 1037:

Fréquence d'échantillonnage:	48 kHz
Résolution de codage:	16 bits par échantillon
Mode:	stéréo

Ligne 2: Le fichier d'origine est enregistré sous forme de fichier BWF linéaire, avec codage MIC, sur l'entrée numérique de la station de travail de réenregistrement, sans dispersion de bruit de quantification:

Fréquence d'échantillonnage:	48 kHz
Résolution de codage:	16 bits par échantillon
Mode:	stéréo

(Champ QualityReport du fragment qualité)

Lignes 1 et 2: codes de sécurité fichier du fragment «qualité» et des données «wave».

Les autres informations sont utilisées selon le processus de saisie sur cassette DAT comme dans l'exemple 1 du § 5.1 ci-dessus.

(Champ CueSheet du fragment qualité)

Les données de montage sonore sont utilisées selon le processus de saisie sur cassette DAT comme dans l'exemple 1 du § 5.1 ci-dessus.