Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

J.203

(11/2006)

SÉRIE J: RÉSEAUX CÂBLÉS ET TRANSMISSION DES SIGNAUX RADIOPHONIQUES, TÉLÉVISUELS ET AUTRES SIGNAUX MULTIMÉDIAS

Application à la télévision numérique interactive

Noyau commun d'une plate-forme d'enregistrement vidéo numérique

Recommandation UIT-T J.203

Recommandation UIT-T J.203

Noyau commun d'une plate-forme d'enregistrement vidéo numérique

Résumé

La présente Recommandation définit les API, les garanties sémantiques et les aspects système de la plate-forme harmonisée d'enregistrement vidéo numérique.

Source

La Recommandation UIT-T J.203 a été approuvée le 29 novembre 2006 par la Commission d'études 9 (2005-2008) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette 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) et considère 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.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux développeurs de consulter la base de données des brevets du TSB sous http://www.itu.int/ITU-T/ipr/.

TABLE DES MATIÈRES

Page

Recommandation UIT-T J.203

Noyau commun d'une plate-forme d'enregistrement vidéo numérique

1 Domaine d'application

La présente Recommandation définit une extension modulaire à [UIT-T J.202], et la mise à jour de GEM [ETSI TS 102 819], qui définit comment l'enregistrement et la lecture de contenus vidéo (et audio) numériques sont intégrés dans la plate-forme GEM [ETSI TS 102 819]. La présente Recommandation est d'abord destinée à être utilisée par des entités qui écrivent les spécifications et/ou normes de terminaux qui étendent une spécification de terminal GEM [ETSI TS 102 819] avec un enregistrement et une lecture de vidéo (et audio) numérique. Elle est ensuite destinée aux développeurs d'applications GEM [ETSI TS 102 819] qui souhaitent utiliser l'enregistrement et la lecture de vidéo (et audio) numérique. Les développeurs devraient consulter les éditeurs des spécifications qui se réfèrent à GEM [ETSI TS 102 819] en ce qui concerne la conformité.

NOTE – La présente Recommandation définit les interfaces visibles des applications. Les développeurs d'application ne devraient pas supposer qu'une interface est disponible tant qu'elle ne figure pas spécifiquement sur la liste. Les normes ou implémentations de terminal peuvent avoir d'autres interfaces présentes. Un des principaux objectifs de la présente Recommandation est de maximiser les aspects communs en ce qui concerne l'intégration de l'enregistrement vidéo/audio numérique entre MHP [ETSI ES 201 812] et les diverses spécifications de terminaux GEM [ETSI TS 102 819].

2 Références

2.1 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée. La référence à un document figurant dans la présente Recommandation ne donne pas à ce document, en tant que tel, le statut d'une Recommandation.

[UIT-T J.202] Recommandation UIT-T J.202 (2005), *Harmonisation des formats de contenus procéduraux pour les applications de télévision interactive.*

[ETSI ES 201 812] ETSI ES 201 812 V1.1.1: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3" *(Vidéodiffusion numérique (DVB); Spécification 1.0.3 de la plate-forme multimédia résidentielle (MPH)).*

[ETSI TS 102 812] ETSI TS 102 812 V1.3.1: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.2" *(Vidéodiffusion numérique (DVB); Spécification 1.1.2 de la plate-forme multimédia résidentielle (MPH)).*

[ETSI TS 102 819] ETSI TS 102 819: "Digital Video Broadcasting (DVB); Globally Executable MHP (GEM)" *(Vidéodiffusion numérique (DVB); Plate-forme multimédia résidentielle exécutable dans le monde entier (GEM)).*

2.2 Références informatives

Les références suivantes sont fournies à titre d'information afin de ne pas restreindre l'application de la présente Recommandation aux services de TV-Anytime, mais les utilisateurs sont invités à utiliser, lorsque c'est possible, les normes ci-dessous, aux fins d'harmonisation.

– ETSI TS 102 822-1 V1.3.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 1: Benchmark Features" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 1: Caractéristiques de référencement).*

– ETSI TS 102 822-2 V1.3.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 2: System description" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 2: Description du système).*

– ETSI TS 102 822-3-1 V1.3.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 – Metadata schemas" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 3: Métadonnées; sous-partie 1: Phase 1 – schémas de métadonnées).*

– ETSI TS 102 822-3-2 V1.3.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 2: System aspects in a uni-directional environment" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 3: Métadonnées; sous-partie 2: Aspects système dans un environnement unidirectionnel).*

– ETSI TS 102 822-3-3 V1.1.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 3: Phase 2 – Extended Metadata Schema" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 3: Métadonnées; sous-partie 3: Phase 2 – Schéma étendu de métadonnées).*

– ETSI TS 102 822-3-4 V1.1.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 4: Phase 2 – Interstitial metadata" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 3: Métadonnées; sous-partie 4: Phase 2 – Métadonnées interstitielles).*

– ETSI TS 102 822-4 V1.2.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 4: Content referencing" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 4: Référencement de contenu).*

– ETSI TS 102 822-5 V1.1.1 (2005-03): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1"); Part 5: Rights Management and Protection (RMP) Information for Broadcast Applications" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 5: Informations de gestion et protection des droits pour les applications en diffusion).*

– ETSI TS 102 822-5-1 V1.2.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 5: Rights Management and Protection (RMP) Sub-part 1: Information for Broadcast Applications" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur*

les systèmes de mémorisation personnels ("TV-Anytime"); Partie 5: Gestion et protection des droits; Sous-partie 1: Informations pour les applications en diffusion).

– ETSI TS 102 822-5-2 V1.2.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 5: Rights Management and Protection (RMP) Sub-part 2: RMPI binding" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 5: Gestion et protection des droits; Sous-partie 2: Liaisons des informations de gestion et protection des droits).*

– ETSI TS 102 822-6-1 V1.3.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 6: Delivery of metadata over a bi-directional network; Sub-part 1: Service and transport" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 6: Livraison des métadonnées sur un réseau bi-directionnel; Sous-partie 1: Service et transport).*

– ETSI TS 102 822-6-2 V1.3.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 6: Delivery of metadata over a bi-directional network; Sub-part 2: Phase 1 – Service discovery" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 6: Livraison des métadonnées sur un réseau bi-directionnel; Sous-partie 2: Phase 1 – Découverte du service).*

– ETSI TS 102 822-6-3 V1.1.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 6: Delivery of metadata over a bi-directional network; Sub-part 3: Phase 2 – Exchange of Personal Profile" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 6: Livraison des métadonnées sur un réseau bi-directionnel; Sous-partie 3: Phase 2 – Echange de profil personnel).*

– ETSI TS 102 822-7 V1.1.1 (2003-10): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1"); Part 7: Bi-directional metadata delivery protection" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 7: Protection bi-directionnelle de la livraison des métadonnées).*

– ETSI TS 102 822-8 V1.1.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 8: Phase 2 – Interchange Data Format" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 8: Phase 2 – Format d'échange des données).*

– ETSI TS 102 822-9 V1.1.1 (2006-01): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 9: Phase 2 – Remote Programming" *(Services en diffusion et en ligne: Recherche, choix et utilisation correcte du contenu sur les systèmes de mémorisation personnels ("TV-Anytime"); Partie 9: Phase 2 – Programmation à distance).*

3 Définitions

La présente Recommandation définit les termes suivants:

3.1 spécification de GEM pour l'enregistrement: spécification complète qui étend la présente Recommandation pour créer une spécification complète sur la façon d'intégrer l'enregistrement et la lecture vidéo (et audio) numérique dans les spécifications de terminaux GEM.

3.2 terminal d'enregistrement de GEM: terminal ou autre appareil qui se conforme à la spécification de GEM pour l'enregistrement.

3.3 lecture normale: passage d'un contenu vidéo et/ou audio vers l'avant à une vitesse de 1,0. Cela inclut les contenus en direct, les contenus en direct avec décalage temporel et les contenus qui sont le résultat d'un enregistrement programmé.

3.4 flux enregistrables: flux de données au sein d'un contenu à enregistrer et reproduire. Pour les spécifications de terminaux GEM, cela inclut des flux identifiés correspondant aux § 7.2.1 ("Audio") et 7.2.2 ("Vidéo") de GEM [ETSI TS 102 819]. Les spécifications de GEM pour l'enregistrement peuvent définir d'autres types de flux comme flux enregistrables comme des sous-titres ou des closed-captions.

3.5 enregistrement: terme générique qui se réfère à un concept profane d'événement programmé ou enregistré, et ne se réfère pas à un objet Java réel. Le terme recouvre des enregistrements qui ont été demandés mais n'ont pas encore débuté, des enregistrements qui sont en cours, des enregistrements qui sont terminés (réussis ou non) et des enregistrements qui ont échoué sans qu'aucune donnée n'ait été enregistrée.

3.6 enregistrements programmés: enregistrements dont la cible d'enregistrement est spécifiée soit par l'heure, la durée et le canal ou par un identifiant qui se résout en heure, durée et canal.

3.7 chronologie de synthèse: chronologie pour une partie de contenu qui a été synthétisée par le terminal d'enregistrement GEM (par opposition à une inclusion au titre de la partie de contenu lors de la transmission).

3.8 chronologie: avancement conceptuel dans le temps inhérent à un élément de contenu, auquel peuvent se référer les applications et qui est délivré par une chronologie de transmission.

3.9 enregistrements programmés: enregistrements où la cible de l'enregistrement est du contenu vidéo et audio en cours de réception.

3.10 chronologie transmise: c'est une chronologie incluse au titre d'un morceau de contenu lorsqu'un contenu est transmis (par exemple, mode d'avancement normal DSM-CC).

3.11 mode artificiel: terme générique pour la lecture de vidéo et/ou audio à des vitesses autres que 1,0 ou des directions autres que vers l'avant. Cela inclut des vitesses de lecture accélérées ou ralenties à la fois vers l'avant et vers l'arrière ainsi que la pause.

3.12 application à détection de mode artificiel: application signalée comme étant capable de détecter une lecture en mode artificiel, c'est-à-dire, avec le fanion trick_mode_aware_flag mis à '1'.

4 Abréviations

La présente Recommandation utilise les abréviations suivantes:

AIT tableau d'informations d'application (*application information table*), tel que défini au § 11.4 de [ETSI ES 201 812] et [ETSI TS 102 812].

DVR magnétoscope numérique (*digital video recorder*)

PDR enregistreur numérique personnel (*personal digital recorder*)

5 Conventions

Le terme "application" est utilisé à la fois pour les applications GEM et les applications qui sans être elles-mêmes des applications GEM sont conformes à la spécification de terminal GEM implémentée par un terminal d'enregistrement GEM particulier.

La présente Recommandation utilise les expressions disant que quelque chose "ne doit pas être enregistré" ou "ne devrait pas être enregistré". C'est une convention dans un souci de concision, et

dans tous les cas, les implémentations sont autorisées à enregistrer des éléments qui ne doivent pas ou ne devraient pas "être enregistrés" et à les supprimer lors de la lecture. Le terme "ne doit pas être enregistré" est simplement plus court que "ne doit pas être enregistré, ou s'il est enregistré, doit être supprimé durant la lecture" qui est ce que cela signifie en réalité.

6 Généralités

6.1 Objet

La présente Recommandation n'est pas destinée à être, et ne devrait pas être utilisée comme étant, une spécification complète de la façon dont on doit intégrer un enregistrement et une reproduction de vidéo (et audio) numérique dans les spécifications de terminal GEM. Elle est un cadre de travail sur lequel peut être créé une telle spécification complète (appelée la spécification de GEM pour l'enregistrement).

6.2 Conformité à la présente Recommandation

Tableau 6-1 – Spécifications de terminal MHP et GEM

[ETSI ES 201 812]	MHP 1.0
[ETSI TS 102 812]	MHP 1.1
[ETSI TS 102 819]	GEM

Tableau 6-2 – Spécifications de MHP et GEM pour l'enregistrement

[b-ETSI TS 102 816]	Extension PVR/PDR à la plate-forme multimédia résidentielle
[b-OC-SP-OCAP-DVR-I02-050524]	Magnétoscope numérique OCAP

Pour éviter les doutes, les équipements qui sont entièrement conformes à la totalité de la présente Recommandation sauf au paragraphe ci-dessus ne sont pas totalement conformes à la présente Recommandation.

7 Processus d'enregistrement et de lecture

Les implémentations de la présente Recommandation doivent effectuer ce qui suit au titre de leur processus d'enregistrement et de lecture.

7.1 Gestion des enregistrements programmés

Le processus de gestion des enregistrements programmés inclut les activités suivantes:

1) maintenir une liste des demandes d'enregistrements qui restent en état d'attente au moins jusqu'à la fin de la période de validité de la demande d'enregistrement;
2) maintenir une liste des demandes d'enregistrements qui ont été terminés avec succès, de celles dont l'enregistrement a commencé mais n'a pas pu être mené à bien, et de celles dont l'enregistrement était programmé mais dont le lancement a échoué;
3) initialiser le processus d'enregistrement pour une demande d'enregistrement en cours à l'heure appropriée, y compris le réveil à partir d'un état de gestion d'attente ou similaire si c'est nécessaire;
4) maintenir des références aux demandes d'enregistrements qui ont échoué dans la liste des demandes d'enregistrement pendant au moins la période de validité de la demande d'enregistrement.

NOTE – Les mécanismes de résolution des conflits entre les demandes d'enregistrements (par exemple, l'utilisation du tuner) sont en dehors du domaine d'application de la présente Recommandation et devraient être spécifiés par les spécifications de GEM pour l'enregistrement.

7.2 Le processus d'enregistrement

Le processus d'enregistrement doit inclure les activités suivantes:

1) Identifier quels flux enregistrables devraient être enregistrés. Si des flux spécifiques avaient été spécifiés lors de la programmation originelle de la demande d'enregistrement, ce sont ceux là qui devraient être enregistrés. Si aucun flux spécifique n'avait été spécifié lors de la programmation originelle de la demande d'enregistrement, les flux par défaut de la partie de contenu devraient être enregistrés.

   NOTE 1 – La définition des flux par défaut à enregistrer est en dehors du domaine d'application de la présente Recommandation et devrait être spécifiée par les spécifications de GEM pour l'enregistrement.

2) L'enregistrement des flux identifiés, jusqu'à la limite de la capacité d'enregistrement du terminal d'enregistrement GEM. Lorsque devraient être enregistrés plus de flux de tout type que ce que peut enregistrer le terminal d'enregistrement GEM, les flux doivent être classés par priorité, conformément au § 11.4 de [ETSI TS 102 819] (qui se réfère à son tour au § 11.4.2.3 de [ETSI ES 201 812] et [ETSI TS 102 812]).

   NOTE 2 – Les capacités minimales du nombre de flux de chaque type que les terminaux d'enregistrement GEM doivent être capables d'enregistrer sont en dehors du domaine d'application de la présente Recommandation et devraient être spécifiées par les spécifications de GEM pour l'enregistrement.

3) Identifier les applications enregistrables et les enregistrer avec des données suffisantes pour reconstruire leurs entrées de tableau AIT.

   Les applications avec une description d'enregistrement d'application où le fanion scheduled_recording_flag *(fanion d'enregistrement programmé)* est mis à 0 ne doivent pas être considérées comme enregistrables.

   NOTE 3 – Une définition plus complète des applications qui sont enregistrables (et de celles qui ne le sont pas) est en dehors du domaine d'application de la présente Recommandation et devrait être spécifiée par les spécifications de GEM pour l'enregistrement.

   NOTE 4 – Les exigences pour qu'un terminal d'enregistrement GEM surveille les données dynamiques et le comportement des applications durant un enregistrement sont en dehors du domaine d'application de la présente Recommandation et devraient être spécifiées par les spécifications de GEM pour l'enregistrement.

4) Enregistrer des informations suffisantes sur toutes les chronologies transmises qui font partie de l'enregistrement afin de leur permettre d'être correctement reconstruites durant la lecture.

5) Générer un débit de support qui s'incrémente de façon linéaire au taux de 1,0 depuis le début jusqu'à la fin de l'enregistrement.

6) Traiter les cas suivants qui se rapportent à l'interruption d'alimentation électrique d'un terminal d'enregistrement GEM durant un enregistrement programmé – l'enregistrement est en cours et l'alimentation est coupée – l'enregistrement débute alors que l'alimentation n'est pas disponible mais l'alimentation revient avant la fin de l'enregistrement programmé – l'enregistrement est en cours et l'alimentation est coupée et revient une ou plusieurs fois – l'enregistrement débute alors que l'alimentation n'est pas disponible mais l'alimentation revient puis est coupée une ou plusieurs fois. Dans tous ces cas, on doit enregistrer autant de contenu que ce qui est disponible alors que le terminal d'enregistrement GEM a été alimenté.

7.3 Gestion des enregistrements achevés

Le processus de gestion des enregistrements achevés doit inclure les activités suivantes:

1) maintenir avec tous les enregistrements achevés les informations suivantes tant que le contenu est conservé;
   - si l'enregistrement est achevé ou incomplet ou si cette information n'est pas connue;
   - l'heure et le canal d'enregistrement;
   - les données spécifiques de l'application associées à l'enregistrement;
2) supprimer éventuellement l'enregistrement (y compris l'entrée correspondante dans la liste des enregistrements, les données enregistrées et toutes autres informations associées) une fois passée la période d'expiration pour la demande d'enregistrement du feuillet correspondant à cet enregistrement.

   NOTE – La présente Recommandation ne parle pas de la précision avec laquelle la période d'expiration devrait être respectée. Les spécifications de GEM pour l'enregistrement devraient spécifier avec quelle précision elle doit être mise en application par les implémentations.

7.4 Reproduction d'enregistrements programmés

7.4.1 Processus de lecture

Le processus de lecture d'enregistrements programmés doit inclure les activités suivantes:

1) commencer la lecture des flux enregistrables sauf si le fanion initiating_replay_flag *(fanion d'initialisation de lecture)* dans la description d'enregistrement d'application est mis à "1";
2) commencer la lecture des applications enregistrées lorsqu'elles font partie de l'élément de contenu enregistré et que leur code application_control_code *(code de contrôle d'application)* est AUTOSTART;

   NOTE – Les exigences sur la reconstruction de comportement dynamique des applications enregistrées durant la lecture sont en dehors du domaine d'application de la présente Recommandation et devraient être spécifiées par les spécifications de GEM pour l'enregistrement.
3) lors de l'exécution du contenu qui est en cours d'enregistrement, si la fin du contenu à enregistrer est atteinte et que l'enregistrement s'arrête, la lecture doit se poursuivre sans interruption (mais pas nécessairement sans raccord), indépendamment de tout processus (dépendant de l'implémentation) pour copier le contenu nouvellement enregistré à partir d'une mémoire tampon temporaire sur un support plus permanent de l'appareil de stockage;
4) on doit synthétiser une heure qui augmente de façon linéaire depuis le début jusqu'à la fin du contenu enregistré. Elle doit être utilisée comme base de "l'heure de base de temps" et de "temps de support" tels que définis par JMF. Aucune relation n'est exigée entre cette heure et toute heure qui fait partie du contenu enregistré, comme MPEG PCR, STC ou DSMCC NPT;
5) pour toutes les lignes d'heure transmises qui font partie de l'enregistrement original, on doit reconstruire chaque ligne de temps lorsque le temps de support en cours est dans la gamme pour laquelle cette ligne de temps est valide.

7.4.2 Evénements durant la lecture

Durant la lecture du contenu enregistré à la suite d'un enregistrement programmé, le comportement suivant soit être pris en charge:

Tableau 7-1 – Evénements durant la lecture normale et comportement résultant

Evénement	Comportement	Résultat à l'écran
Avance rapide à la fin du flux	Evénement de fin de support généré sur toute application enregistrée	Dernière trame gelée
Retour au début du flux	Passage en mode pause	Première trame gelée
Avance en fin de flux	Fin de support générée sur toute application enregistrée	Dernière trame gelée

7.5 Programmation en différé

7.5.1 Processus d'enregistrement

Le processus de la programmation d'enregistrement en différé doit inclure les activités suivantes:

1) Identifier quels sont les flux enregistrables à enregistrer et les enregistrer.

   NOTE 1 – La définition des flux à enregistrer en différé est en dehors du domaine d'application de la présente Recommandation et devrait être spécifiée par les spécifications de GEM pour l'enregistrement.

2) Identifier les applications enregistrables signalées avec le contenu enregistré au début de l'enregistrement et enregistrer aux moins ces applications et des données suffisantes pour reconstruire leurs entrées de tableau d'AIT.

   NOTE 2 – La définition des applications enregistrables en différé (et de celles qui ne le sont pas) est en dehors du domaine d'application de la présente Recommandation et devrait être spécifiée par les spécifications de GEM pour l'enregistrement.

   NOTE 3 – L'exigence qu'un terminal d'enregistrement GEM surveille les données dynamiques et le comportement des applications durant un enregistrement différé est en dehors du domaine d'application de la présente Recommandation et devrait être spécifiée par les spécifications de GEM pour l'enregistrement.

3) Identifier les lignes horaires transmises qui font partie du contenu en cours d'enregistrement et enregistrer sur elles des informations suffisantes pour leur permettre d'être reconstruites précisément durant la lecture.

4) Gérer la mémoire tampon de programmation différée de telle sorte que lorsque cette mémoire tampon est pleine, l'enregistrement se poursuive avec recouvrement progressif du contenu restant le plus ancien de la mémoire tampon.

7.5.2 Lecture

La lecture du contenu enregistré en mode différé exige qu'une mémoire tampon de programmation différée soit associée à un lecteur JMF ou un contexte de service. La définition de la façon et du moment où cette association est faite est en dehors du domaine d'application de la présente Recommandation et devrait être spécifiée par les spécifications de GEM pour l'enregistrement.

Durant la lecture du contenu enregistré en mode différé, le comportement doit être celui défini au § 7.4 avec les modifications suivantes:

1) Le comportement défini par le Tableau 7-1: Evénements durant la lecture normale et comportement résultant, doit être remplacé par le tableau ci-dessous.

Tableau 7-2 – Evénements durant la lecture en différé et comportement résultant

Evénement	Comportement
Avance rapide à la fin de la mémoire tampon de différé (c'est-à-dire, au point où s'écrit le contenu qui s'enregistre)	Passer en lecture au taux de 1,0 au sein de l'enregistrement différé ou en dire et sans détruire les contenus de la mémoire tampon.
Retour au début de la mémoire tampon de différé (c'est-à-dire, au point où le contenu enregistré est réécrit)	Passer en lecture normale au taux de 1,0 au sein de l'enregistrement mis en mémoire depuis le début du contenu de la mémoire de différé.
Lecture en fin de mémoire tampon de différé (c'est-à-dire, qu'une mauvaise réception du signal cause une sous alimentation en données dans la mémoire de différé)	Ne rien faire (c'est-à-dire, continuer la lecture au taux de 1,0 au sein de l'enregistrement mis en mémoire) ou passer au direct sans détruire les contenus de la mémoire tampon de différé.
Pause jusqu'à ce que la mémoire de différé soit "pleine" et que le contenu au point où la pause est faite soit près d'être écrasé	Passer à lecture au taux de 1,0 depuis le point où la pause a été faite.

Pour toutes les chronologies transmises qui sont valides dans la mémoire tampon de différé, reconstruire chaque ligne temporelle lorsque le temps de support en cours est dans la gamme pour laquelle cette chronologie est valide.

2) Le temps de support pour chaque localisation dans la mémoire tampon de différé doit être la valeur allouée à cette localisation au point où il a été enregistré. Il ne doit pas y avoir de discontinuités dans le temps de support au sein de l'enregistrement mis en mémoire, y compris la "tête" de la mémoire tampon où le contenu est le même que dans le contenu qui vient d'être reçu.

3) Régler le temps de support à une valeur correspondant à POSITIVE_INFINITY *(infini positif)* doit placer la localisation de lecture au point d'enregistrement en cours si l'enregistrement est toujours en cours, ou, s'il ne l'est pas, à la fin de l'enregistrement.

4) Si la localisation de lecture est la même que le point d'enregistrement (c'est-à-dire, le moment présent est en cours de lecture), le contenu doit être affiché avec un délai de zéro par rapport au contenu diffusé original. Les implémentations qui peuvent afficher le contenu enregistré avec un délai de zéro par rapport au direct peuvent afficher le contenu enregistré. Les autres implémentations doivent afficher le contenu de diffusion original.

8 API d'enregistrement et de lecture

8.1 Enregistrement et gestion de l'enregistrement

8.1.1 Généralités (pour information)

Les API d'enregistrement permettent aux applications d'effectuer des enregistrements de deux façons différentes:

1) programmer la réalisation d'un enregistrement à un moment donné futur;

2) enregistrer des événements diffusés en temps réel. Cela peut aussi inclure la portion de l'événement de diffusion qui se trouve dans une mémoire tampon d'enregistrement différé.

Les implémentations entretiennent une base de données des enregistrements, représentée par le singleton *(élément unique)* RecordingManager *(gestionnaire d'enregistrement)*. Les applications créent des enregistrements en appelant la méthode record() de l'objet RecordingManager. Cette opération a pour effet de créer une entrée dans la base de données d'enregistrement entretenue par l'implémentation. Ces entrées sont représentées par des instances de RecordingRequest *(demandes d'enregistrement)*. Les applications peuvent associer des données spécifiques de l'application à des demandes RecordingRequest en utilisant la méthode addAppData(). Ces données spécifiques de

l'application pourraient être des descriptions d'événement dans un format privé d'application ou une clé dans une base de données d'informations entretenue par l'application.

Lors de la création d'un enregistrement, les applications spécifient un ensemble de propriétés d'enregistrement. Une d'elles est une période d'expiration mesurée à partir du moment de l'enregistrement. Comme défini au § 7.3 ci-dessus, une fois passé ce délai d'expiration, l'implémentation va supprimer l'enregistrement. Une seconde propriété est la période de validité mesurée à partir du moment de la demande d'enregistrement. Si l'enregistrement n'a pas été fait au moment de la fin de cette période, l'enregistrement peut être supprimé.

Les applications peuvent acquérir un ensemble d'enregistrements représentés par une RecordingList *(liste d'enregistrements)*. Les applications qui souhaitent limiter l'ensemble des enregistrements dans une RecordingList peuvent l'exprimer au moyen d'un RecordingListFilter *(filtre de liste d'enregistrements)*. La présente Recommandation définit certains filtres qui sont disponibles pour les applications. Les spécifications de GEM pour l'enregistrement peuvent définir leurs propres filtres. Les applications peuvent créer leurs propres filtres. Les filtres peuvent être mis en cascade de sorte que la sortie d'un filtre soit utilisée comme l'entrée d'un filtre suivant.

NOTE – Les spécifications d'enregistrement GEM peuvent définir des règles gouvernant l'accès des enregistrements aux applications. Lorsque ceci est fait, l'API qui en fait la liste ne retournera pas les enregistrements auxquels l'application appelante n'est pas autorisée à accéder selon ces règles.

Lorsque des données commencent à être enregistrées pour un enregistrement, un RecordedService est créé. RecordedService étend l'interface de service JavaTV et forme le lien entre l'API d'enregistrement et l'API de lecture.

8.1.2 Détails

Les paquetages org.ocap.shared.dvr et org.ocap.shared.dvr.navigation doivent être pris en charge.

Dans la spécification ServiceContextRecordingSpec, les spécifications de GEM pour l'enregistrement peuvent rendre obligatoires des caractéristiques normalement facultatives de mémorisation et d'enregistrement des contenus de la mémoire tampon de différé lorsque l'heure de début startTime est dans le passé.

Lorsqu'un enregistrement programmé débute au moment où l'appareil terminal d'enregistrement GEM est mis hors tension, mais que l'appareil est mis sous tension alors que l'enregistrement programmé est toujours valable, l'implémentation doit exécuter les étapes suivantes:

1) Régler l'état de la demande LeafRecordingRequest conformément aux ressources disponibles.
2) Si des ressources sont disponibles, créer un RecordedService et commencer un enregistrement associé.
3) Si l'enregistrement se termine, régler l'état LeafRecordingRequest à INCOMPLETE_STATE *(état inachevé)*.
4) Régler l'Exception à retourner avec la méthode LeafRecordingRequest.getFailedException à org.ocap.shared.dvr.RecordingFailedException.POWER_INTERRUPTION.

Lorsqu'un enregistrement programmé est en cours et que l'alimentation du terminal d'enregistrement GEM est coupée, mais que l'appareil est remis sous tension et que le contenu à enregistrer n'est pas achevé, l'implémentation doit exécuter les étapes suivantes:

1) Créer une demande LeafRecordingRequest pour l'enregistrement restant aussitôt que le processus d'amorce s'achève et que l'implémentation détecte qu'un enregistrement programmé devrait être en cours. Copier les données spécifiques de l'application à partir de la demande LeafRecordingRequest originale sur celui qui vient d'être créé. Régler l'état conformément aux ressources disponibles. Régler la RecordingSpec pour qu'elle corresponde à la demande d'enregistrement originale.

2) Si des ressources sont disponibles, créer un RecordedService et commencer l'enregistrement associé.

3) Si l'enregistrement se termine, régler l'état de la demande LeafRecordingRequest à INCOMPLETE_STATE.

4) Régler l'Exception à retourner par la méthode LeafRecordingRequest.getFailedException à org.ocap.shared.dvr.RecordingFailedException.POWER_INTERRUPTION.

Si l'alimentation est coupée puis rétablie sur le terminal d'enregistrement GEM plusieurs fois pendant la durée d'un enregistrement programmé, une demande LeafRecordingRequest et un RecordedService doivent être créés à chaque fois que l'alimentation est rétablie et que des ressources sont disponibles pour commencer à enregistrer.

NOTE 1 – Les spécifications de GEM pour l'enregistrement peuvent ajouter des exigences supplémentaires pour combiner des parties d'enregistrement programmé interrompues par une perte d'alimentation en une seule construction (par exemple, une sous-classe de RecordingRequest) leur permettant ainsi d'être lues et manipulées comme une seule entité.

Lorsqu'un enregistrement programmé est en cours et que l'alimentation du terminal d'enregistrement GEM est coupée, mais que l'appareil est réalimenté alors que le contenu à enregistrer est terminé, l'implémentation doit exécuter les étapes suivantes:

1) Régler l'état de la demande LeafRecordingRequest à INCOMPLETE_STATE.

2) Régler l'Exception à retourner par la méthode LeafRecordingRequest.getFailedException à INSUFFICIENT_RESOURCES.

Lorsqu'un enregistrement programmé est en cours et que l'alimentation est coupée au terminal d'enregistrement GEM, mais que l'appareil est réalimenté alors que le contenu à enregistrer n'est PAS terminé, l'implémentation doit exécuter les étapes suivantes:

1) Créer une demande LeafRecordingRequest pour l'enregistrement restant aussitôt le processus d'amorçage terminé et que l'implémentation détecte qu'un enregistrement programmé devrait être en cours. Copier les données spécifiques de l'application à partir de la demande LeafRecordingRequest originelle sur celle qui vient d'être créée. Régler l'état conformément aux ressources disponibles. Régler la RecordingSpec de façon à correspondre à la demande d'enregistrement originelle.

2) Si des ressources sont disponibles, créer un RecordedService et commencer l'enregistrement associé.

3) Si l'enregistrement se termine, régler l'état LeafRecordingRequest à INCOMPLETE_STATE.

4) Régler l'Exception à retourner par la méthode LeafRecordingRequest.getFailedException à org.ocap.shared.dvr.RecordingFailedException.POWER_INTERRUPTION.

Si l'alimentation est coupée et rétablie plusieurs fois au terminal d'enregistrement GEM pendant la durée d'un enregistrement programmé, une demande LeafRecordingRequest et un RecordedService doivent être créés à chaque fois que l'alimentation a été rétablie et que des ressources sont disponibles pour commencer l'enregistrement.

NOTE 2 – Les spécifications de GEM pour l'enregistrement peuvent ajouter des exigences supplémentaires pour combiner des parties d'un enregistrement programmé interrompu par une perte d'alimentation en une seule construction (par exemple, une sous-classe de RecordingRequest) et donc leur permettant d'être lues et manipulées comme une seule entité.

8.2 Lecture

8.2.1 Généralités (pour information)

La lecture d'un enregistrement peut être effectuée de deux façons, selon qu'il est souhaité ou non de lancer toutes les applications qui pourraient faire partie de l'enregistrement.

1) Appeler la méthode de select(Service) d'un ServiceContext passant dans le RecordedService à lire sélectionnera tous les composants de service du service y compris toutes applications à lancement automatique qui auraient été enregistrées.

2) Appeler la méthode getMediaLocator() sur un RecordedService et passer le résultat à javax.media.Manager.createPlayer(MediaLocator). Une fois qu'un lecteur a été obtenu, start() ou syncStart() commencera la lecture. Cela lira seulement les flux enregistrables au sein de l'enregistrement et aucune des applications.

Une fois que la lecture a commencé, on peut la contrôler d'une des façons suivantes:

1) utiliser les commandes JMF retournées de Player.getControl(String) ou Player.getControls() pour des caractéristiques telles que le choix du langage audio et l'étalonnage de la vidéo;

2) utiliser Player.setRate(float) pour contrôler la vitesse de lecture, y compris de changer de la lecture à vitesse normale, l'avance rapide, le retour et la pause.

8.2.2 Détails

Le paquetage org.ocap.shared.media doit être pris en charge. Les lecteurs JMF qui présentent les contenus d'une mémoire tampon de différé doivent prendre en charge la commande TimeshiftControl pour ce paquetage.

Les extensions suivantes doivent être appliquées aux API exigées par [ETSI TS 102 819]:

1) La méthode javax.tv.service.selection.ServiceContext.select(Service) doit accepter les instances de RecordedService comme étant des entrées valides. Lorsqu'il est appelé avec une telle instance, le contenu enregistré de ce RecordedService doit être choisi dans le ServiceContext spécifié.

2) La méthode javax.media.Manager.createPlayer(MediaLocator) doit accepter les mediaLocators retournés par RecordedService.getMediaLocator comme étant une entrée valide et retourner un lecteur JMF. Lorsqu'un tel lecteur entre dans l'état démarré, les flux enregistrables de ce RecordedService doivent être présentés.

3) Lorsqu'un RecordedService est choisi avec succès dans un ServiceContext, la base de données AppsDatabase doit être remplie à partir de l'ensemble des applications enregistrées au titre de ce RecordedService comme si ces applications étaient transmises en direct.

   NOTE 1 – AppsDatabase devrait être mise à jour dans la mesure où la spécification de GEM pour l'enregistrement exige des changements de surveillance et de reconstruction dans la signalisation de l'application.

4) Durant la lecture des flux enregistrables (aussi bien la lecture d'un enregistrement programmé que l'enregistrement en différé), lorsque le débit de lecture change, un événement javax.media.RateChangeEvent doit être envoyé à toutes les applications avec des ControllerListeners enregistrés sur un lecteur JMF pour ce contenu.

5) Les appels à la méthode setRate d'un lecteur JMF doivent contrôler la vitesse de lecture, y compris les changements entre lecture à vitesse normale, en avance rapide, en retour rapide et en pause.

6) Les appels à la méthode Player.setMediaTime doivent essayer de commencer la présentation du contenu aussi proche que possible du temps de support spécifié, sauf s'il s'agit d'un temps de support passé retourné par MediaTimeFactoryControl. Les

approximations SetTimeApproximations de la sémantique définie par cette méthode doivent être observées.

7) Lorsque un RecordedService est en cours de lecture, les appels à la méthode ServiceDomain.attach(..) avec le localisateur retourné par RecordedService.getLocator doivent fonctionner comme spécifié et donner l'accès au système de fichier en diffusion dans l'enregistrement. La présente Recommandation ne définit pas d'exigence pour l'accès aux systèmes de fichiers en diffusion dans l'enregistrement lorsque cet enregistrement n'est pas en lecture; cependant de telles exigences peuvent être définies par les spécifications de GEM pour l'enregistrement.

Lorsque les lecteurs JMF présentent les flux enregistrables d'un RecordedService ou les contenus de la mémoire tampon de différé, les commandes JMF suivantes (définies par [ETSI TS 102 819] ou la présente Recommandation ou comme spécifié explicitement) doivent au minimum être prises en charge.

- org.davic.media.AudioLanguageControl
- org.davic.media.FreezeControl
- javax.tv.media.MediaSelectControl
- org.ocap.shared.media.TimeLineControl
- org.ocap.shared.media.TimeFactoryControl
- org.davic.media.MediaTimeEventControl comme défini dans [b-DAVIC 1.4.1p9].

NOTE 2 – Les spécifications de GEM pour l'enregistrement peuvent exiger que soient prises en charge les commandes JMF pour RecordedServices ou les contenus d'une mémoire tampon de différé. Différents ensembles de commandes JMF peuvent être spécifiés pour ces deux cas.

8.3 Autres API

8.3.1 Questions relatives à l'état des versions

Les propriétés dont la liste figure dans les deux tableaux suivants doivent être incluses dans l'ensemble des propriétés de la classe java.lang.System. Et donc, ces propriétés peuvent être restituées en utilisant java.lang.System.getProperty(). Comme cette API retourne une chaîne, les valeurs numériques retournées doivent être codées comme défini par java.lang.Integer.toString(int).

Tableau 8-1 – Propriétés système pour l'interrogation de version

Propriété	Sémantique	Valeurs possibles	Exemple
gem.recording.version.major	Numéro de version majeur de la version de la présente Recommandation prise en charge.	Valeur d'entier non négatif	"1"
gem.recording.version.minor	Numéro de version mineur de la version de la présente Recommandation prise en charge.	Valeur d'entier non négatif	"0"
gem.recording.version.micro	Numéro de version micro de la version de la présente Recommandation prise en charge.	Valeur d'entier non négatif	"0"

8.4 Permissions

8.4.1 Applications non signées

Depuis RecordingPermission, seule la RecordingPermission("read", "own") doit être accordée à des applications non signées.

8.4.2 Applications signées

Lorsque le fichier de demande de permission demande la permission de créer une demande d'enregistrement et qu'elle est accordée, une RecordingPermission("create", "own") est créée.

Lorsque le fichier de demande de permission demande la permission de modifier une demande d'enregistrement et qu'elle est accordée, une RecordingPermission("modify", "own") est créée.

Lorsque le fichier de demande de permission demande la permission de supprimer une demande d'enregistrement et qu'elle est accordée, une RecordingPermission("delete", "own") est créée.

Lorsque le fichier de demande de permission demande la permission d'annuler une demande d'enregistrement et qu'elle est accordée, une RecordingPermission("cancel", "own") est créée.

Les spécifications de GEM pour l'enregistrement peuvent définir un mécanisme pour permettre aux applications d'avoir des instances de RecordingPermission dont la chaîne d'action est "*" mais ce n'est pas exigé.

9 Signalisation d'application

9.1 Description d'enregistrement d'applications

Les spécifications de GEM pour l'enregistrement doivent définir une description d'enregistrement d'application suffisante pour déduire ce qui suit:

Tableau 9-1 – Description d'enregistrement d'application

Fonction	Type
scheduled_recording__flag	booléen
Trick_mode_aware_flag	booléen
Time_shift_flag	booléen
initiating_replay_flag	booléen
label_count	entier non signé
for(i=0;i<N0;i++){	
label_length	entier non signé de 8 bits
for(j=0;i<N1;i++){	
label_payload	entier non signé de 8 bits
}	
storage_properties	entier non signé de 2 bits
}	

scheduled_recording_flag: Ce fanion d'un seul bit, lorsqu'il est mis à '1', indique que l'application est appropriée pour enregistrer lorsque le service dans lequel il est signalé est enregistré par un enregistrement programmé. Lorsqu'il est mis à '0', il indique que l'application n'est pas appropriée pour enregistrer avec un enregistrement programmé. Des exemples de la raison pour laquelle une application serait inappropriée pour l'enregistrement sont que l'application n'a pas été testée dans un environnement PDR, ou que l'application est en relation étroite avec l'heure de transmission et qu'il serait absurde pour l'utilisateur final de la lire d'après l'enregistrement (par exemple, une application liée à un événement en direct).

trick_mode_aware_flag: Ce fanion d'un seul bit, lorsqu'il est mis à '1', indique que l'application peut fonctionner en mode artificiel. S'il est mis à '0', l'application n'est pas capable de mode artificiel.

time_shift_flag: Ce fanion d'un seul bit, lorsqu'il est mis à '1', indique que l'application est appropriée à l'enregistrement lorsque le service dans lequel il est signalé est enregistré en mode d'enregistrement en différé. Lorsqu'il est mis à '0', il indique que l'application est inappropriée pour enregistrer en mode différé.

initiating_replay_flag: Ce fanion d'un seul bit, lorsqu'il est mis à '1', indique que le terminal d'enregistrement GEM ne doit pas initialiser la lecture des flux enregistrables sur le même enregistrement que l'application. L'application est responsable du lancement de cette lecture. S'il est mis à '0', l'implémentation doit initialiser cette lecture en parallèle avec le début de l'application comme ce serait le cas conventionnel.

NOTE 1 – Si plusieurs applications sont enregistrées avec un programme ou un service et que ces applications sont signalées comme constituant le point d'entrée du programme de lecture, lorsque les flux enregistrables qui s'y rapportent sont lus, la première application à auto démarrage qui s'y rapporte trouvée dans le tableau d'AIT mémorisé doit être lancée.

label_count: Ce champ de 8 bits identifie le nombre d'étiquettes qui ont été utilisées.

label_length: Ce champ de 8 bits identifie le nombre d'octets dans l'étiquette.

label_char: Ces champs de 8 bits portent un dispositif d'octets qui étiquettent une partie de l'application au sein de son protocole de transport.

NOTE 2 – La présente Recommandation ne définit pas quelles parties des applications peuvent être étiquetées, ou la forme de l'étiquette (s'il en est). L'étiquetage peut être effectué au niveau des fichiers ou des groupes de fichiers ou à des niveaux inférieurs de constructions spécifiques du protocole utilisé pour le transport de l'application.

storage_properties: Champ qui indique l'importance de mémoriser la partie étiquetée de l'application. Les valeurs pour ce champ sont définies comme suit:

0 ne devrait pas être mémorisé

1 mémorisation critique

2 mémorisation facultative

3 réservée

Les spécifications de GEM pour l'enregistrement qui incluent la définition MHP de l'équivalent fonctionnel de "Application Signalling" de GEM doivent satisfaire à cette exigence en prenant en charge le descripteur d'enregistrement d'application défini à l'Annexe A.

10 Modèles d'applications

10.1 Cycle de vie d'application et lecture en mode artificiel

Lorsque la lecture du contenu enregistré a été initialisée en choisissant un RecordedService, le modèle d'application et le cycle de vie doivent être modifiés comme suit:

1) Les applications en cours qui ne sont pas explicitement signalées comme capables de mode artificiel (le fanion trick_mode_aware_flag dans la description d'enregistrement d'application est mis à '1') doivent être arrêtées lorsque le mode de lecture passe de normal à artificiel. Les applications explicitement signalées comme capables de mode artificiel doivent continuer.

2) Lorsque la lecture quitte le mode artificiel et retourne à normal, le terminal d'enregistrement GEM doit évaluer la signalisation de l'application pour ce point dans le contenu et commencer ou arrêter les applications comme nécessaire. Les applications qui commencent doivent être commencées comme si l'utilisateur final venait de passer à une diffusion du contenu concerné.

NOTE – Les spécifications de GEM pour l'enregistrement peuvent permettre des délais dans le redémarrage des applications après le retour en lecture normale si on pense que cela améliore les sensations de l'utilisateur final, par exemple durant des cycles répétés d'avance rapide/lecture/avance rapide/lecture.

Le passage de lecture en direct à lecture en différé ne doit pas automatiquement tuer les applications MHP alors que le contenu différé est lu à partir de la fin de la mémoire tampon de différé, (c'est-à-dire, le point où le contenu nouvellement enregistré est gravé).

11 Sécurité

11.1 Introduction (pour information)

La présente Recommandation inclut deux modèles de sécurité qui commandent la capacité des applications à fonctionner sur des demandes d'enregistrement et des enregistrements terminés.

1) Un modèle se fonde sur l'association d'attributs à des applications MHP/OCAP. Ces attributs sont exprimés sous forme de classes de permission Java. Certains appels de méthode sont protégés et lancent un SecurityException *(exception de sécurité)* si des applications sans permissions spécifiées les appellent. Ce modèle est indépendant des détails de la RecordingRequest concernée.

2) Le second modèle se fonde sur l'association d'attributs de sécurité à des demandes d'enregistrement individuelles. Ces attributs déterminent quelles applications peuvent effectuer quelles opérations sur cette demande d'enregistrement. La présente Recommandation n'évoque pas le mécanisme par lequel ces attributs sont associés à une demande d'enregistrement.

Dans le cas normal, la capacité d'une application à utiliser les caractéristiques fournies par la présente Recommandation est gouvernée par l'intersection des deux ensembles d'attributs. Les opérations sur une RecordingRequest échoueront si l'application appelante n'a pas la permission Java nécessaire pour l'opération et si les attributs associés à la RecordingRequest en cours de traitement ne permettent pas à l'application appelante d'effectuer cette opération. Les spécifications de GEM pour l'enregistrement peuvent définir un mécanisme pour que des applications de confiance totale obtiennent une permission qui les rendent capables de court-circuiter le second modèle et d'avoir le droit d'effectuer toutes les opérations sur toutes les RecordingRequests.

11.2 Fichier de demande de permission

Le DTD du fichier de demande de permission défini par la spécification du terminal GEM doit inclure au moins les éléments et attributs associés suivants:

```
<!ELEMENT recordingpermission EMPTY>
<!ATTLIST recordingpermission
create (true|false) "false"
modify (true|false) "false"
delete (true|false) "false"
cancel (true|false) "false"
>
```

12 Exigences minimales pour le receveur

Les exigences suivantes s'appliquent à tous les terminaux d'enregistrement GEM:

1) Les rapports de vitesse de lecture suivants doivent au moins être pris en charge:

   −16x, −8x, −4x, −2x, −1x, 0, 0,5x, 1x, 2x, 4x, 8x, 16x.

NOTE – Pour le rapport −1x, il est admis d'afficher seulement des i-trames.

Annexe A

Description d'enregistrement d'application

Les spécifications de GEM pour l'enregistrement qui incluent la définition MHP de l'équivalent fonctionnel "Application Signalling" de GEM doivent satisfaire à cette exigence en prenant en charge le descripteur d'enregistrement d'application défini comme suit.

Le descripteur d'enregistrement d'application peut être signalé dans la boucle de descripteur d'application de l'AIT. Ce descripteur contient des informations supplémentaires sur le cycle de vie de l'application, indiquant en particulier si une application est appropriée à une utilisation dans des conditions de lecture en mode artificiel. Il indique si cette application doit ou non être enregistrée lorsqu'un programme, avec lequel cette application est signalée, est enregistré. Il donne le moyen de spécifier les localisations de ressources de données qui doivent être enregistrées avec l'application, ainsi que les étiquettes des modules de carrousel d'objets de l'application qui doivent, devraient ou ne devraient pas être enregistrées.

Tableau A.1 – Syntaxe du descripteur d'enregistrement d'application

Syntaxe	Nombre de bits	Identifiant	Commentaires/ Valeur
application_recording_descriptor (){			
descriptor_tag	8	uimsbf	6
descriptor_length	8	uimsbf	
scheduled_recording_flag	1	bslbf	
trick_mode_aware_flag	1	bslbf	
time_shift_flag	1	bslbf	
dynamic_flag	1	bslbf	
av_synced_flag	1	bslbf	
initiating_replay_flag	1	bslbf	
reserved	7	bslbf	
label_count	8	uimsbf	N0
for(i=0;i<N0;i++){			
label_length	8	uimsbf	N1
for(j=0; j<N1; j++) {			
label_char	8	uimsbf	
}			
storage_properties	2	uimsbf	
Reserved	6		
}			
component_tag_list_length	8	uimsbf	N2
for(i=0;i<N2;i++){			
component_tag	8	uimsbf	
}			
private_length	8	uimsbf	N3
for(i=0;i<N3;i++){			
Private	8	uimsbf	
}			
for(i=0;i<N4;i++){			
reserved_future_use	8	uimsbf	
}			
}			

La sémantique des champs définie dans ce descripteur doit être comme défini ci-dessus au § 9.1 sauf mention contraire dans le présent paragraphe.

descriptor_tag *(étiquette de descripteur)*: Cet entier de 8 bits d'une valeur de 0x06 identifie ce descripteur.

dynamic_flag *(fanion dynamique)*: Ce fanion indique si l'application s'appuie sur l'utilisation de données dynamiques durant son exécution. Lorsqu'il est mis à 1, il indique que l'application s'appuie

sur la présence de fichiers (code ou données) ou de signalisation d'application (par exemple, du code de contrôle d'application) qui changent pendant la durée de vie du morceau de contenu.

NOTE 1 – La présente Recommandation ne définit pas le comportement des terminaux d'enregistrement GEM qui est conditionné par la valeur de ce drapeau. Les spécifications de terminal GEM peuvent utiliser ce drapeau pour déterminer si une application est enregistrable ou non.

av_synced_flag: Ce fanion indique si l'application exige l'utilisation d'événements déclencheurs. S'il est mis à '1', c'est exigé.

NOTE 2 – La présente Recommandation ne définit pas le comportement des terminaux d'enregistrement GEM qui est conditionné par la valeur de ce drapeau. Les spécifications de terminal GEM peuvent utiliser ce fanion pour déterminer si une application est ou non enregistrable.

label_char: Ces champs de 8 bits portent un rang d'octets qui sont une étiquette de module. Cette étiquette correspond à un ou plusieurs modules portés par les descripteurs Label *(étiquette)* dans les champs userInfo *(informations d'utilisateur)* de la structure moduleInfo *(informations de module)* des DII, comme défini au § B.2.2.4.1 de [ETSI ES 201 812] et dans [ETSI TS 102 812].

component_tag_list_length: Cet entier spécifie la longueur en nombre d'octets de la liste des étiquettes de composants.

component_tag: Ce champ identifie un composant de service qui livre les données requises par l'application au moment de la lecture et qui doivent être enregistrées avec l'application. Les composants qui portent des flux enregistrables ont seulement besoin de figurer sur la liste si des composants autres que ceux par défaut doivent être enregistrés. Les composants qui sont des flux portant des carrousels d'objet DSMCC ou des tableaux d'information d'application MHP ne devraient pas figurer sur la liste et devraient être simplement ignorés s'ils y figurent. Excepté cela, les composants qui sont des flux portant des sections privées MPEG-2 devraient figurer sur la liste si leur enregistrement est souhaitable.

private: Ces octets peuvent être utilisés pour des extensions privées.

reserved_future_use: Ces octets réservés pourront être utilisés pour des extensions futures de DVB.

Annexe B

Responsabilités des spécifications de GEM pour l'enregistrement

B.1 Exigences

Ci-après figure une liste d'éléments identifiés dans la présente Recommandation comme étant en dehors de son domaine d'application et que doivent spécifier les spécifications de GEM pour l'enregistrement.

1) Quels types de flux sont à considérer comme des "flux enregistrables". Les types de flux correspondant aux § 7.2.1 ("Audio") et 7.2.2 ("Vidéo") de GEM dans la spécification de terminal GEM sur laquelle se fonde la spécification de GEM pour l'enregistrement doivent être considérés comme des "flux enregistrables".

2) Les mécanismes pour résoudre les conflits entre les enregistrements demandés (par exemple, l'utilisation du tuner).

3) Les capacités minimales en nombre de flux (ou en nombre de chaque type de flux) qu'un terminal d'enregistrement GEM doit être capable d'enregistrer.

4) La définition des applications qui sont enregistrables à la fois en programmé et en différé (qui ne sont pas nécessairement les mêmes).

5) L'exigence qu'un terminal d'enregistrement GEM surveille les données dynamiques (dans le carrousel d'objet DSMCC ou dans l'équivalent fonctionnel GEM) pendant l'enregistrement programmé et en différé (qui ne sont pas nécessairement les mêmes).

6) L'exigence qu'un terminal d'enregistrement GEM surveille les événements de déclenchement GEM ou de flux DSMCC durant l'enregistrement programmé et en différé (qui ne sont pas nécessairement les mêmes).

7) L'exigence qu'un terminal d'enregistrement GEM surveille les signaux d'application dynamiques durant l'enregistrement programmé et en différé (qui ne sont pas nécessairement les mêmes).

8) Les exigences sur la reconstruction de la structuration dans le temps des données dynamiques, des déclencheurs/événements de flux et les signaux d'application dynamiques durant la lecture d'enregistrements programmés et en différé (qui ne sont pas nécessairement les mêmes). Les spécifications de GEM pour l'enregistrement doivent définir des exigences à la fois pour la lecture à vitesse normale et la lecture en mode artificiel.

9) Avec quelle précision la période d'expiration devrait être mise en application par les implémentations.

10) La définition d'au moins un protocole pour les lignes horaires transmises.

11) Les conditions où un lecteur JMF ou un contexte de service est lié à une mémoire tampon de différé.

12) Les exigences sur la taille des données privées spécifiques d'application qu'il doit être possible d'associer à une seule clé avant l'invocation d'un IllegalArgumentException *(exception d'argument illégal).*

13) Les exigences sur le nombre d'entrées de données privées spécifiques d'application qu'il doit être possible d'associer à une seule RecordingRequest avant l'invocation d'une NoMoreDataEntriesException *(exception plus d'entrées de données).*

14) Un mécanisme pour associer des attributs de sécurité à des demandes d'enregistrement individuelles et une transposition de ce mécanisme dans le langage de chacune des méthodes du tableau suivant qui se rapporte aux "attributs de sécurité spécifiques de RecordingRequest".

Tableau B.1 – Méthodes avec incidences sur les attributs de sécurité spécifiques de la demande d'enregistrement

Méthode
`RecordingManager.addRecordingChangedListener(RecordingListListener)`
`RecordingManager.getEntries()`
`RecordingManager.getEntries(RecordingListFilter)`
`RecordingRequest.getRecordingProperties(int)`
`RecordingRequest.add.AppData(int,java.io.Serializable)`
`RecordingRequest.removeAppData(int)`
`RecordingRequest.setRecordingProperties(RecordingSpec)`
`RecordingRequest.delete()`
`RecordingManager.record()`
`RecordingRequest.cancel()`
`RecordingRequest.stop()`
`LeafRecordingRequest.getService()`
`RecordedService.setMediaTime()`

B.2 Facultatif

Ci-après figure une liste d'éléments identifiés dans la présente Recommandation comme étant en dehors de son domaine d'application et que peuvent spécifier les spécifications de GEM pour l'enregistrement.

1) Les mécanismes pour contrôler dans quelle mesure une application peut lire ou modifier des enregistrements programmés et des enregistrements achevés faits par une autre application.
2) Des sous-classes de RecordingListFilter pour filtrer la liste des enregistrements selon des manières non prises en charge par la présente Recommandation.
3) Les règles qui gouvernent l'accès d'une application aux enregistrements.
4) Les commandes JMF supplémentaires qui doivent être prises en charge pour RecordedServices ou les contenus d'une mémoire tampon de différé. Différents ensembles de commandes JMF peuvent être spécifiés pour ces deux cas.
5) Des délais de redémarrage des applications après le retour en lecture normale si on pense que cela va améliorer les sensations de l'utilisateur final, par exemple lors de cycles répétés d'avance rapide/lecture/avance rapide/lecture.
6) Un mécanisme pour permettre à des applications de toute confiance d'obtenir des instances de RecordingPermission dont le paramètre action est "*".
7) Que le comportement facultatif défini dans la description de classe de ServiceContextRecordingSpec, lorsque les contenus de la mémoire tampon de différé sont mémorisés alors que le paramètre startTime est dépassé, devienne obligatoire dans cette spécification particulière de GEM pour l'enregistrement.
8) Les exigences sur la reconstruction de la structuration dans le temps des données dynamiques, des déclencheurs/événements de flux et les signaux d'application dynamiques durant la lecture d'enregistrements programmés et en différé (qui ne sont pas nécessairement les mêmes). Les spécifications de GEM pour l'enregistrement doivent définir des exigences à la fois pour la lecture à vitesse normale et la lecture en mode artificiel.

9) Avec quelle précision la période d'expiration devrait être mise en application par les implémentations.

10) La définition d'au moins un protocole pour les lignes de temps transmises.

11) Les conditions où un lecteur JMF ou un contexte de service est lié à une mémoire tampon de différé.

12) Les exigences sur la taille des données privées spécifiques d'application qu'il doit être possible d'associer à une seule clé avant l'invocation d'un IllegalArgumentException.

13) Les exigences sur le nombre d'entrées de données privées spécifiques d'application qu'il doit être possible d'associer à une seule RecordingRequest avant l'invocation d'une NoMoreDataEntriesException.

14) Un mécanisme pour associer des attributs de sécurité à des demandes d'enregistrement individuelles et une transposition de ce mécanisme dans le langage de chacune des méthodes du Tableau B.1 qui se rapporte aux "attributs de sécurité spécifiques de RecordingRequest".

Annexe C

Références externes; errata, clarifications et exemptions

C.1 Cadre de travail de support Java

C.1.1 javax.media.Clock

Dans la présente Recommandation, le texte suivant de la spécification pour cette classe doit être précisé comme décrit ci-dessous:

La transformation qu'une horloge définit sur une base de temps est définie par trois paramètres: débit *(rate)*, heure de début du support (mst, *media starttime*), et heure de début fondée sur l'heure (tbst, *time-base start-time*). Etant donnée une heure fondée sur l'heure (tbt), l'heure support (mt, *media time*) peut être calculée en utilisant la transformation suivante:

$$mt = mst + (tbt – tbst)*rate$$

Le débit (*rate*) est un simple facteur d'échelle qui est appliqué à la base de temps. Par exemple, un débit de 2,0 indique que l'horloge va tourner deux fois plus vite que sa base de temps. De même, un débit négatif indique que l'horloge tourne dans la direction opposée à celle de sa base de temps.

L'heure de début fondée sur l'heure et l'heure de début du support définissent un point commun dans le temps auquel l'horloge et la base de temps sont synchronisées.

Chaque changement de débit réussit doit être considéré comme étant "un point commun dans le temps auquel sont synchronisées l'horloge et la base de temps" comme défini ci-dessus. Les valeurs de l'horloge et de la base de temps au point commun doivent être les valeurs à l'instant du changement de débit.

Annexe D

Paquetages API d'un noyau commun d'une plate-forme d'enregistrement vidéo numérique

D.1 Paquetage d'enregistrement video numérique partagé

```
org.ocap.shared.dvr
     Package
```


D.1.1 Classe Service_RecordingSpec

```
org.ocap.shared.dvr
     Class ServiceRecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
       +--org.ocap.shared.dvr.ServiceRecordingSpec
```

---

classe publique **ServiceRecordingSpec**

étend RecordingSpec

Spécifie une demande d'enregistrement en termes de Service.

Lorsque des instances de cette classe sont passées à RecordingManager.record(..), le mode d'échec supplémentaire suivant doit s'appliquer – si l'heure de fin (calculée comme étant l'heure de début plus la durée) est passée lorsque la méthode d'enregistrement est invoquée, la méthode d'enregistrement doit lancer une IllegalArgumentException.

Lorsqu'une instance de cette spécification d'enregistrement est passée comme paramètre à la méthode RecordingRequest.reschedule(..), une IllegalArgumentException est lancée si la source est différente de celle spécifiée dans la spécification d'enregistrement en cours pour la demande d'enregistrement et si celle-ci est dans l'état en progrès *(progress)*.

Lorsque des instances de cette classe sont passées à RecordingManager.record(..), si l'heure de début est passée et si

– aucun des contenus concernés n'est déjà enregistré; ou

– certains des contenus concernés sont déjà enregistrés mais que l'implémentation ne prend pas en charge l'inclusion de contenus déjà enregistrés dans un enregistrement programmé;

alors l'heure en cours doit être utilisée comme heure de début, et la durée doit être réduite en conséquence. La présente Recommandation n'exige pas que les implémentations incluent de contenu déjà enregistré dans des enregistrements programmés, mais les spécifications de GEM pour l'enregistrement peuvent l'exiger.

Résumé constructeur
`ServiceRecordingSpec(javax.tv.service.Service source, java.util.Date startTime, long duration, RecordingProperties properties)` **Constructeur.**

Résumé pour la méthode	
`long`	**`getDuration`**`()` Retourne la durée passée comme argument au constructeur.
`javax.tv.service.Service`	**`getSource`**`()` Retourne la source de l'enregistrement
`java.util.Date`	**`getStartTime`**`()` Retourne l'heure de début passée comme argument au constructeur.

Méthodes héritées de la classe org.ocap.shared.dvr.RecordingSpec
`getProperties`

Méthodes héritées de la classe java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Détails du constructeur

ServiceRecordingSpec

```
public ServiceRecordingSpec(javax.tv.service.Service source,
                            java.util.Date startTime,
                            long duration,
                            RecordingProperties properties)
                     throws java.lang.IllegalArgumentException
```

Constructeur.

Paramètres:

`source` – Le service à enregistrer

`startTime` – Heure de début de l'enregistrement

`duration` – Durée à enregistrer en millisecondes

`properties` – Définition de la façon dont l'enregistrement doit être réalisé.

Invocation:

`java.lang.IllegalArgumentException` – si la source n'est pas un service de diffusion ou si la durée est négative

Détails de méthode

getSource

```
public javax.tv.service.Service getSource()
```

Retourne la source de l'enregistrement

Retourne:

la source passée au constructeur

---

getStartTime

```
public java.util.Date getStartTime()
```

Retourne l'heure de début passée comme argument au constructeur.

Retourne:

l'heure de début passée dans le constructeur

---

getDuration

```
public long getDuration()
```

Retourne la durée passée comme argument au constructeur.

Retourne:

la durée passée dans le constructeur

D.1.2 Classe AccessDeniedException

```
org.ocap.shared.dvr
Class AccessDeniedException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.dvr.AccessDeniedException
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **AccessDeniedException**

étend java.lang.Exception

Exception invoquée lorsque le fonctionnement d'une application est bloqué sur une demande RecordingRequest par des attributs de sécurité associés à cette RecordingRequest.

Voir aussi:

Forme en série

Résumé du constructeur
AccessDeniedException() Construit une AccessDeniedException sans message d'explication

Méthodes héritées de la classe java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détails du constructeur

AccessDeniedException

public **AccessDeniedException**()

Construit une AccessDeniedException sans message d'explication

D.1.3 Classe DeletionDetails

```
org.ocap.shared.dvr
Class DeletionDetails

java.lang.Object
  +--org.ocap.shared.dvr.DeletionDetails
```

classe publique **DeletionDetails**

étend java.lang.Object

Cette classe contient des détails sur la suppression d'un service enregistré. Des instances de cette classe sont construites par l'implémentation et retournées aux applications à partir de la méthode `getDeletionDetails`.

Résumé de champ	
`static int`	**`EXPIRED`** Code de cause: Le service enregistré a été supprimé par l'implémentation à cause de l'arrivée à expiration de la demande d'enregistrement.
`static int`	**`USER_DELETED`** Code de cause: Le service enregistré a été supprimé explicitement par l'application.

Résumé du constructeur
`DeletionDetails(int reason, java.util.Date date)` Construit des DeletionDetails

Résumé de méthode	
`java.util.Date`	**`getDeletionTime`**`()` Donne la date et l'heure à laquelle le service enregistré a été supprimé.
`int`	**`getReason`**`()` Rapporte la raison de la suppression du service enregistré.

Méthodes héritées de la classe java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Détail du champ

`EXPIRED`

`public static final int` **`EXPIRED`**

Code de cause: Le service enregistré a été supprimé par l'implémentation parce que la demande d'enregistrement est arrivée à expiration.

Voir aussi:

Valeurs de champ constantes

USER_DELETED

```
public static final int USER_DELETED
```

Code de cause: Le service enregistré a été explicitement supprimé par l'application.

Voir aussi:

Valeurs de champ constantes

Détail pour le constructeur

DeletionDetails

```
public DeletionDetails(int reason,
                       java.util.Date date)
```

Construit un DeletionDetails

Paramètres:

`reason` – cause de la suppression du service enregistré

`date` – date et heure de la suppression du service enregistré

Détail pour la méthode

getReason

```
public int getReason()
```

Rapporte la cause de la suppression du service enregistré. C'est la valeur telle que passée par le constructeur.

Retourne:

Le code de cause de la suppression du service enregistré.

---

getDeletionTime

```
public java.util.Date getDeletionTime()
```

Obtient la date et l'heure de la suppression du service enregistré. C'est la valeur telle qu'elle est passée au constructeur.

Retourne:

La date et l'heure de la suppression.

D.1.4 Interface LeafRecordingRequest

```
org.ocap.shared.dvr
        Interface LeafRecordingRequest
```

Toutes les super interfaces:
RecordingRequest

---

interface publique **LeafRecordingRequest**

étend RecordingRequest

Cette interface représente des informations correspondant à une demande d'enregistrement de niveau final. La demande d'enregistrement représentée par cette interface correspond à une demande d'enregistrement qui a été entièrement résolue en un seul enregistrement.

Une demande d'enregistrement d'extrémité peut être en instance *(pending)* (c'est-à-dire en attente de l'heure de début), en cours *(in-progress)*, terminée *(completed)*, inachevée *(incompleted)*, ou échouée *(failed)*.

Lorsqu'elle est dans l'état en instance, une demande d'enregistrement peut entrer en conflit avec d'autres enregistrements pour les ressources. De tels conflits doivent être résolus avant l'heure de début programmée pour l'enregistrement, sinon, la demande d'enregistrement en cours devra résulter en un enregistrement échoué.

Résumé de champ	
`static int`	**`COMPLETED_STATE`** L'enregistrement pour cette demande d'enregistrement s'est achevé avec succès.
`static int`	**`DELETED_STATE`** Le service enregistré correspondant à cette demande d'enregistrement a été supprimé.
`static int`	**`FAILED_STATE`** La demande d'enregistrement a échoué.
`static int`	**`IN_PROGRESS_INSUFFICIENT_SPACE_STATE`** L'enregistrement pour cette demande d'enregistrement a été initialisé et est en cours, mais l'implémentation a détecté que l'espace de mémorisation pourrait n'être pas suffisant pour achever l'enregistrement.
`static int`	**`IN_PROGRESS_STATE`** L'enregistrement pour cette demande d'enregistrement a été initialisé et est en cours.
`static int`	**`INCOMPLETE_STATE`** L'enregistrement pour cette demande d'enregistrement a été initialisé mais a échoué dans le IN_PROGRESS_STATE avant d'avoir pu atteindre le COMPLETED_STATE.
`static int`	**`PENDING_NO_CONFLICT_STATE`** La demande d'enregistrement est en instance. L'enregistrement pour cette demande est supposé s'achever avec succès.
`static int`	**`PENDING_WITH_CONFLICT_STATE`** La demande d'enregistrement pourrait n'être pas initialisée à cause de conflits de ressources.

Résumé de méthode	
`void`	**`cancel`**`()` Annule une demande d'enregistrement en instance.
`DeletionDetails`	**`getDeletionDetails`**`()` Obtient des informations détaillées sur la suppression du service enregistré correspondant à cette demande d'enregistrement.

java.lang.Exception	**getFailedException**() Obtient l'exception qui a causé l'entrée de la demande d'enregistrement dans l'état FAILED_STATE, ou INCOMPLETE_STATE.
RecordedService	**getService**() Retourne le RecordedService correspondant à la demande d'enregistrement.
void	**stop**() Arrête l'enregistrement pour une demande d'enregistrement en cours indépendamment de la durée qui a été enregistrée.

Méthodes héritées de l'interface org.ocap.shared.dvr.RecordingRequest

addAppData, delete, getAppData, getAppID, getId, getKeys, getParent, getRecordingSpec, getRoot, getState, isRoot, removeAppData, reschedule, setRecordingProperties

Détail du champ

PENDING_NO_CONFLICT_STATE

public static final int **PENDING_NO_CONFLICT_STATE**

La demande d'enregistrement est en instance. L'enregistrement pour cette demande est supposé s'achever avec succès.

Voir aussi:

Valeurs de champ constantes

PENDING_WITH_CONFLICT_STATE

public static final int **PENDING_WITH_CONFLICT_STATE**

La demande d'enregistrement peut ne pas être intialisée du fait de conflits de ressources. L'implémentation a détecté un conflit de ressources pour l'heure programmée de cette demande d'enregistrement et la résolution actuelle du conflit ne permet pas que cette demande d'enregistrement soit initialisée avec succès.

Voir aussi:

Valeurs de champ constantes

IN_PROGRESS_STATE

public static final int **IN_PROGRESS_STATE**

L'enregistrement a été initialisé pour cette demande d'enregistrement et est en cours. L'enregistrement est supposé s'achever avec succès.

Voir aussi:

Valeurs de champ constantes

IN_PROGRESS_INSUFFICIENT_SPACE_STATE

```
public static final int IN_PROGRESS_INSUFFICIENT_SPACE_STATE
```

L'enregistrement a été initialisé pour cette demande d'enregistrement et est en cours, mais l'implémentation a détecté que l'espace de mémorisation pourrait n'être pas suffisant pour achever l'enregistrement. Cette situation peut survenir lorsque l'implémentation détecte que l'espace mémoire pourrait n'être pas suffisant pour achever cette demande d'enregistrement et que d'autres demandes d'enregistrement sont en cours. La demande d'enregistrement n'est pas supposée pouvoir être menée à bien. Une demande d'enregistrement peut entrer dans cet état à partir de IN_PROGRESS_STATE ou PENDING_NO_CONFLICT_STATE. Une implémentation peut commencer une demande d'enregistrement en IN_PROGRESS_STATE sur la base d'une estimation initiale de l'espace nécessaire et passer ultérieurement à IN_PROGRESS_INSUFFICIENT_SPACE_STATE lorsque elle a suffisamment d'informations pour calculer une estimation plus précise de l'espace nécessaire. Si une implémentation ne peut pas détecter à l'avance un espace insuffisant, l'implémentation peut commencer la demande d'enregistrement en IN_PROGRESS_STATE puis passer ensuite à FAILED_STATE lorsque l'espace manque. Il est aussi possible qu'une demande d'enregistrement commence en IN_PROGRESS_INSUFFICIENT_SPACE_STATE et passe ensuite à IN_PROGRESS_STATE ou COMPLETED_STATE. Cela peut arriver si d'autres demandes d'enregistrement sont supprimées après le début de l'enregistrement ou si l'implémentation calcule une meilleure estimation de l'espace nécessaire au cours de l'avancement de l'enregistrement.

Voir aussi:

Valeurs de champ constantes

INCOMPLETE_STATE

```
public static final int INCOMPLETE_STATE
```

L'enregistrement pour cette demande d'enregistrement a débuté mais a échoué dans l'état IN_PROGRESS_STATE avant d'avoir pu atteindre l'état COMPLETED_STATE. La demande RecordingRequest contiendra un `RecordedService` qui peut être lu quelle que soit la durée enregistrée.

Voir aussi:

Valeurs de champ constantes

COMPLETED_STATE

```
public static final int COMPLETED_STATE
```

L'enregistrement pour cette demande d'enregistrement s'est achevé avec succès.

Voir aussi:

Valeurs de champ constantes

FAILED_STATE

```
public static final int FAILED_STATE
```

La demande d'enregistrement a échoué.

Voir aussi:

Valeurs de champ constantes

DELETED_STATE

```
public static final int DELETED_STATE
```

Le service enregistré correspondant à cette demande d'enregistrement a été supprimé.

Voir aussi:

Valeurs de champ constantes

Détail pour la méthode

cancel

```
public void cancel()
            throws java.lang.IllegalStateException,
                   AccessDeniedException
```

Annule une demande d'enregistrement en instance. La demande d'enregistrement sera supprimée de la base de données après le succès de l'invocation de cette méthode. Annuler une demande d'enregistrement peut résoudre un ou plusieurs conflits. Dans ce cas, certains enregistrements en instance en conflit passeraient à l'état en instance sans conflit.

Invocation:

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération par des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas RecordingPermission("cancel",..) ou RecordingPermission("*",..).

`java.lang.IllegalStateException` – si l'état de l'enregistrement n'est pas en PENDING_STATE_NO_CONFLICT_STATE ou PENDING_WITH_CONFLICT_STATE.

---

stop

```
public void stop()
          throws java.lang.IllegalStateException,
                 AccessDeniedException
```

Arrêter l'enregistrement pour une demande d'enregistrement en instance indépendamment de la durée qui a été enregistrée. Passer l'enregistrement à l'état INCOMPLETE_STATE.

Invocation:

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération par les attributs de sécurité spécifiques de la demande RecordingRequest.

java.lang.SecurityException – si l'application appelante n'a pas RecordingPermission("cancel",..) ou RecordingPermission("*",..)

java.lang.IllegalStateException – si l'enregistrement n'est pas dans l'état IN_PROGRESS_STATE, ou IN_PROGRESS_INSUFFICIENT_SPACE_STATE.

---

getFailedException

```
public java.lang.Exception getFailedException()
                                    throws java.lang.IllegalStateException
```

Obtient l'exception qui a causé l'entrée de la demande d'enregistrement dans l'état FAILED_STATE, ou INCOMPLETE_STATE.

Retourne:

L'exception qui a causé l'échec. L'exception retournée sera une RecordingFailedException.

Invocation:

java.lang.IllegalStateException – si la demande d'enregistrement n'est pas dans l'état FAILED_STATE ou INCOMPLETE_STATE.

---

getService

```
public RecordedService getService()
                           throws java.lang.IllegalStateException,
                                  AccessDeniedException
```

Retourne le RecordedService correspondant à la demande d'enregistrement.

Retourne:

Le service enregistré associé à la demande d'enregistrement.

Invocation:

java.lang.IllegalStateException – si la demande d'enregistrement n'est pas dans l'état INCOMPLETE_STATE, IN_PROGRESS_STATE, IN_PROGRESS_INSUFFICIENT_SPACE_STATE ou COMPLETED_STATE.

AccessDeniedException – si l'application appelante n'est pas autorisée à effectuer cette opération par des attributs de sécurité spécifiques de la demande RecordingRequest.

---

getDeletionDetails

```
public DeletionDetails getDeletionDetails()
                                  throws java.lang.IllegalStateException
```

Obtient des informations détaillées sur la suppression du service enregistré correspondant à cette demande d'enregistrement.

Retourne:

Les détails de la suppression pour cette demande d'enregistrement.

Invocation:

java.lang.IllegalStateException – si la demande d'enregistrement n'est pas dans l'état DELETED_STATE.

D.1.5 Classe LocatorRecordingSpec

```
org.ocap.shared.dvr
Class LocatorRecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
       +--org.ocap.shared.dvr.LocatorRecordingSpec
```

---

classe publique **LocatorRecordingSpec**

étend RecordingSpec

Spécifie une demande d'enregistrement en termes d'un ou plusieurs localisateurs.

Si plusieurs localisateurs sont contenus au sein de la source, tous DOIVENT faire partie du même service.

Lorsque des instances de cette classe sont passées à RecordingManager.record(..), le mode d'échec supplémentaire suivant doit être appliqué – si l'heure de fin (calculée à heure de début + durée) est passée lorsque la méthode d'enregistrement est invoquée, la méthode d'enregistrement doit lancer une IllegalArgumentException.

Lorsque une instance de cette spécification d'enregistrement est passée comme un paramètre à la méthode RecordingRequest.reschedule(..), une IllegalArgumentException est lancée si la source est différente de la source spécifiée dans la spécification d'enregistrement en cours pour la demande d'enregistrement et si la demande d'enregistrement est dans l'état en cours.

Lorsque des instances de cette classe sont passées à RecordingManager.record(..), si l'heure de début est passée et si

– aucun des contenus concernés n'est déjà enregistré;

– certains des contenus concernés sont déjà enregistrés mais que l'implémentation ne prend pas en charge d'inclure des contenus déjà enregistrés dans un enregistrement programmé;

l'heure courante doit être utilisée comme heure de début et la durée doit être réduite en conséquence. La présente Recommandation n'exige pas des implémentations d'inclure des contenus déjà enregistrés dans des enregistrements programmés, cependant, les spécifications de GEM pour l'enregistrement peuvent l'exiger.

Résumé pour le constructeur
LocatorRecordingSpec(javax.tv.locator.Locator[] source, java.util.Date startTime, long duration, RecordingProperties properties) Constructeur

Résumé pour la méthode	
long	**getDuration**() Retourne la durée passée comme argument au constructeur.
Javax.tv.locator.Locator[]	**getSource**() Retourne la source de l'enregistrement.
java.util.Date	**getStartTime**() Retourne l'heure de début passée comme argument au constructeur.

Méthodes héritées de la classe org.ocap.shared.dvr.RecordingSpec
getProperties

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Détail pour le constructeur

LocatorRecordingSpec

```
public LocatorRecordingSpec(javax.tv.locator.Locator[] source,
                            java.util.Date startTime,
                            long duration,
                            RecordingProperties properties)
                     throws
javax.tv.service.selection.InvalidServiceComponentException
```

Constructeur

Paramètres:

`source` – Source des flux à enregistrer. Les implémentations mises en œuvre doivent faire une copie de ce dispositif avant que le constructeur le retourne.

`startTime` – Heure de début de l'enregistrement.

`duration` – Durée à enregistrer en millisecondes.

`properties` – Définition de la façon d'effectuer l'enregistrement.

Invocation:

`javax.tv.service.selection.InvalidServiceComponentException` – si tous les localisateurs dans le paramètre de source ne sont pas dans le même service.

`java.lang.IllegalArgumentException` – si la durée est négative.

Détail pour la méthode

getSource

```
public javax.tv.locator.Locator[] getSource()
```

Retourne la source de l'enregistrement.

Retourne:

la source passée dans le constructeur

---

getStartTime

```
public java.util.Date getStartTime()
```

Retourne l'heure de début passée comme argument au constructeur.

Retourne:

l'heure de début passée dans le constructeur

---

getDuration

```
public long getDuration()
```

Retourne la durée passée comme argument au constructeur.

Retourne:

la durée passée dans le constructeur

D.1.6 Classe NoMoreDataEntitiesException

```
org.ocap.shared.dvr
        Class NoMoreDataEntriesException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.dvr.NoMoreDataEntriesException
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **NoMoreDataEntriesException**

étend java.lang.Exception

Plus d'entrées de données permises pour cette demande d'enregistrement.

Voir aussi:

Forme en série

Résumé pour le constructeur
NoMoreDataEntriesException() Construit une exception NoMoreDataEntriesException sans message de détail

Méthodes héritées de la classe java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail pour le constructeur

NoMoreDataEntriesException

public **NoMoreDataEntriesException**()

Construit une exception NoMoreDataEntriesException sans message de détail

D.1.7 Interface ParentRecordingRequest

org.ocap.shared.dvr
Interface ParentRecordingRequest

Toutes les super interfaces:

RecordingRequest

---

interface publique **ParentRecordingRequest**

étend RecordingRequest

Cette interface représente les informations correspondant à une demande d'enregistrement parente. La demande d'enregistrement représentée par cette interface peut avoir une ou plusieurs demandes d'enregistrement filles.

Une demande d'enregistrement parente peut être dans l'état non résolu, l'état partiellement résolu, l'état complètement résolu, ou dans l'état annulé.

Une demande d'enregistrement serait dans l'état non résolu si l'implémentation n'a pas assez d'informations pour traiter cette demande d'enregistrement. Une demande d'enregistrement dans cet état peut passer à l'état partiellement résolu, à l'état complètement résolu, où à l'état d'échec.

Une demande d'enregistrement serait dans l'état partiellement résolu si l'implémentation a assez d'informations pour programmer certaines demandes d'enregistrement filles correspondant à cette demande d'enregistrement, mais pas toutes. Cela serait le cas d'une demande d'enregistrement pour une série où certains épisodes de la série sont programmés. Une demande d'enregistrement est dans l'état complètement résolu si tous ses enregistrements fils sont connus et programmés.

Résumé du champ	
static int	**CANCELLED_STATE** Une demande d'enregistrement est dans l'état annulé, si une application a appelé avec succès la méthode d'annulation pour cette demande d'enregistrement, mais que toutes les demandes d'enregistrement filles n'ont pas été supprimées.
static int	**COMPLETELY_RESOLVED_STATE** Tous les enregistrements fils correspondant à cette demande d'enregistrement ont été programmés.
Static int	**PARTIALLY_RESOLVED_STATE** Certains enregistrements correspondant à cette demande d'enregistrement ont été programmés.
Static int	**UNRESOLVED_STATE** L'implémentation n'a pas assez d'informations pour traiter cette demande d'enregistrement.

Résumé pour la méthode	
void	**cancel**() Annule la demande d'enregistrement parente.
RecordingList	**getKnownChildren**() Obtient tous les enregistrements fils correspondant à cette demande RecordingRequest parente.

Méthodes héritées de l'interface org.ocap.shared.dvr.RecordingRequest
addAppData, delete, getAppData, getAppID, getId, getKeys, getParent, getRecordingSpec, getRoot, getState, isRoot, removeAppData, reschedule, setRecordingProperties

Détail de champ

UNRESOLVED_STATE

```
public static final int UNRESOLVED_STATE
```

L'implémentation n'a pas assez d'informations pour traiter cette demande d'enregistrement.

Voir aussi:

Valeurs de champ constantes

COMPLETELY_RESOLVED_STATE

```
public static final int COMPLETELY_RESOLVED_STATE
```

Tous les enregistrements fils correspondant à cette demande d'enregistrement ont été programmés. Une demande d'enregistrement est dans l'état complètement résolu si

l'implémentation a assez d'informations pour programmer les enregistrements correspondants à la demande d'enregistrement. Une demande d'enregistrement en état complètement résolu aurait un ou plusieurs enregistrements fils.

Voir aussi:

Valeurs de champ constantes

PARTIALLY_RESOLVED_STATE

```
public static final int PARTIALLY_RESOLVED_STATE
```

Certains des enregistrements correspondant à cette demande d'enregistrement ont été programmés. Une demande d'enregistrement est dans l'état partiellement résolu si l'implémentation a assez d'informations pour programmer certains des enregistrements correspondants à la demande d'enregistrement, mais pas tous. Une demande d'enregistrement en état partiellement résolu peut avoir zéro, un ou plusieurs enregistrements fils.

Voir aussi:

Valeurs de champ constantes

CANCELLED_STATE

```
public static final int CANCELLED_STATE
```

Une demande d'enregistrement est dans l'état annulé si une application a appelé avec succès la méthode d'annulation pour cette demande d'enregistrement, mais que toutes les demandes d'enregistrement filles n'ont pas été supprimées. Une demande d'enregistrement dans cet état doit être supprimée par l'implémentation une fois que toutes les demandes d'enregistrement filles ont été supprimées.

Voir aussi:

Valeurs de champ constantes

Détail pour la méthode

cancel

```
public void cancel()
            throws java.lang.IllegalStateException,
                   AccessDeniedException
```

Annule la demande d'enregistrement parente. L' implémentation doit aussi annuler toutes les demandes d'enregistrement filles en invoquant leur méthode cancel(). Aucun autre enregistrement fils ne doit être programmé pour cette demande d'enregistrement ou pour aucun de ses enregistrements fils. La demande d'enregistrement sera supprimée de la base de données une fois que toutes les demandes d'enregistrement filles auront été supprimées. Annuler une demande d'enregistrement parente ne supprime aucun des enregistrements fils qui ne peut être annulé (c'est-à-dire une demande d'enregistrement fille qui n'est pas dans l'état en instance). À l'achèvement réussi de cette méthode, la demande d'enregistrement sera supprimée de la base de données ou passée à l'état CANCELLED_STATE.

Invocation:

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("cancel",..) ou de RecordingPermission("*",..)

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.IllegalStateException` – si l'état de la demande d'enregistrement n'est pas UNRESOLVED_STATE, PARTIALLY_RESOLVED_STATE ou COMPLETELY_RESOLVED_STATE.

---

getKnownChildren

```
public RecordingList getKnownChildren()
                              throws java.lang.IllegalStateException
```

Obtient tous les enregistrements fils correspondant à cette demande RecordingRequest parente. Pour une demande d'enregistrement en état complètement résolu, cette méthode retourne tous les enfants qui sont encore conservés dans la base de données de gestion des enregistrements (c'est-à-dire que les enfants retirés de la base de données par l'invocation de la méthode delete() ou cancel() ne seront pas inclus dans la liste des enregistrements fils). Pour une demande d'enregistrement en état partiellement résolu, cette méthode ne retourne que les enfants actuellement connus pour la série.

Retourne:

La liste des enregistrements fils correspondants à cet enregistrement; vide si il n'y a pas de demande d'enregistrement fille dans la base de données RecordingManager.

Invocation:

`java.lang.IllegalStateException` – si la demande d'enregistrement n'est pas dans l'état PARTIALLY_RESOLVED_STATE ou COMPLETELY_RESOLVED_STATE.

D.1.8 Interface RecordedService

```
org.ocap.shared.dvr
       Interface RecordedService
```

Toutes les superinterfaces:

javax.tv.service.Service

---

interface publique **RecordedService**

étend javax.tv.service.Service

Cette interface représente la portion enregistrée d'un service qui est en cours d'enregistrement ou a été enregistré pendant une certaine durée. Aussitôt que commence l'enregistrement est créé un service enregistré. Si la demande d'enregistrement échoue pour une raison quelconque, l'enregistrement doit s'arrêter normalement et le service enregistré doit être disponible et contenir ce qu'il a été enregistré du service.

Un RecordedService a un attribut heure du support qui détermine où commence la lecture lorsque le service enregistré est sélectionné sur un ServiceContext. Cette heure du support est persistante et est applicable pour tous les choix futurs de service par toutes les applications jusqu'à ce que l'heure du support soit changée avec une invocation de setMediaTime.

Noter le comportement spécifique de sous-interface suivant pour les méthodes définies par la super interface javax.tv.service.Service:

– La méthode hasMultipleInstances() doit toujours retourner faux.

– La méthode getServiceType() doit toujours retourner RecordedServiceType.RECORDED_SERVICE.

– La méthode getLocator() doit toujours retourner un localisateur correspondant au service enregistré qui soit différent du localisateur du service d'origine. Ce localisateur devrait retourner ce RecordedService lorsqu'il est entré dans le SIManager.getService(..).

– La méthode getName() doit retourner une chaîne lisible par l'homme.

– Les services enregistrés ne doivent pas être inclus dans les listes de service retournées par la méthode SIManager.filterServices(..).

Résumé pour la méthode	
`void`	**`delete`**`()` Supprime le service enregistré.
`javax.media.Time`	**`getFirstMediaTime`**`()` Obtient l'heure de support JMF au début du service enregistré.
`Javax.media.MediaLocator`	**`getMediaLocator`**`()` Retourne le MediaLocator correspondant au RecordedService.
`javax.media.Time`	**`getMediaTime`**`()` Obtient l'heure de support JMF qui a été établie en utilisant la méthode setMediaTime(..)
`long`	**`getRecordedDuration`**`()` Obtient la durée réelle du contenu enregistré.
`RecordingRequest`	**`getRecordingRequest`**`()` Obtient la RecordingRequest correspondant au RecordedService.
`java.util.Date`	**`getRecordingStartTime`**`()` Obtient l'heure à laquelle a commencé l'enregistrement.
`void`	**`setMediaTime`**`(javax.media.Time mediaTime)` Etablit l'heure de support JMF pour la localisation d'où va commencer la lecture lorsque ce service enregistré est choisi sur un ServiceContext.

Méthodes héritées de l'interface javax.tv.service.Service
`equals, getLocator, getName, getServiceType, hashCode, hasMultipleInstances, retrieveDetails`

Détail pour la méthode

getRecordingRequest

```
public RecordingRequest getRecordingRequest()
```

Obtient la `RecordingRequest` correspondant au RecordedService.

Retourne:

La demande `RecordingRequest` pour le service.

---

getRecordedDuration

```
public long getRecordedDuration()
```

Obtient la durée réelle du contenu enregistré. Pour les enregistrements en cours, cela retournera la durée de la partie achevée de l'enregistrement.

Retourne:

La durée de l'enregistrement en millisecondes.

---

getMediaLocator

```
public javax.media.MediaLocator getMediaLocator()
```

Retourne le MediaLocator correspondant au RecordedService.

Retourne:

RecordedService MediaLocator.

---

setMediaTime

```
public void setMediaTime(javax.media.Time mediaTime)
                  throws AccessDeniedException
```

Règle l'heure de support JMF pour la localisation de l'endroit où commencera la lecture lorsque ce service enregistré est choisi dans un ServiceContext.

Si une instance d'heure correspondant à une valeur de 0 nanoseconde, ou une valeur négative est établie, ou si cette méthode n'a pas été invoquée pour ce service enregistré, la lecture va commencer au début du contenu enregistré. Si l'instance d'heure établie correspond à l'infini positif ou à une valeur supérieure à la durée du contenu enregistré, la lecture va commencer au point en direct si l'enregistrement est en cours pour le service enregistré. Si l'enregistrement n'est pas en cours, la lecture va passer immédiatement à la fin du support. NOTE – Le réglage de l'heure du support sera applicable à tous les futurs choix de service de toutes les applications.

Paramètres:

`mediaTime` – l'heure du support correspondant à la localisation à partir de laquelle va commencer la lecture lorsque ce service est choisi.

Invocation:

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest correspondant à la demande RecordingRequest associée à ce service enregistré.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("modify",..) ou RecordingPermission("*",..)

getMediaTime

```
public javax.media.Time getMediaTime()
```

Obtient l'heure de support JMF qui a été établie en utilisant la méthode setMediaTime(..)

Retourne:

la valeur de l'heure de support JMF qui a été établie en utilisant la méthode setMediaTime(..), si cette méthode a été invoquée auparavant sur ce RecordedService. Si la méthode setMediaTime n'a pas été invoquée auparavant, cette méthode devrait retourner l'heure de support JMF correspondant au début du RecordedService.

---

getRecordingStartTime

```
public java.util.Date getRecordingStartTime()
```

Obtient l'heure d'initialisation de l'enregistrement.

Retourne:

l'heure à laquelle l'enregistrement a été initialisé par l'implémentation.

---

delete

```
public void delete()
            throws AccessDeniedException
```

Supprime le service enregistré. La méthode supprime le service enregistré et tous les flux élémentaires enregistrés (par exemple, les fichiers et les entrées de répertoire) associés au RecordedService. La demande d'enregistrement correspondante va passer à l'état DELETED_STATE.

Si la demande d'enregistrement est dans l'état IN_PROGRESS, l'implémentation va arrêter d'enregistrer avant de supprimer le service enregistré. Si le RecordedService était en cours de présentation lors de la suppression, un `PresentationTerminatedEvent` sera envoyé avec la cause SERVICE_VANISHED.

Invocation:

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("delete",..) ou RecordingPermission("*",..)

---

getFirstMediaTime

```
public javax.media.Time getFirstMediaTime()
```

Obtient l'heure de support JMF au début du service enregistré.

Retourne:

une heure du support.

D.1.9 Classe RecordedServiceType

```
org.ocap.shared.dvr
        Class RecordedServiceType

java.lang.Object
  +--javax.tv.service.ServiceType
       +--org.ocap.shared.dvr.RecordedServiceType
```

---

classe publique **RecordedServiceType**

étend javax.tv.service.ServiceType

Cette classe représente la valeur du type de service pour un RecordedService.

Résumé du champ	
`static javax.tv.service.ServiceType`	**`RECORDED_SERVICE_TYPE`** ServiceType pour un Recorded service.

Champs hérités de la classe javax.tv.service.ServiceType
`ANALOG_RADIO, ANALOG_TV, DATA_APPLICATION, DATA_BROADCAST, DIGITAL_RADIO, DIGITAL_TV, NVOD_REFERENCE, NVOD_TIME_SHIFTED, UNKNOWN`

Résumé pour le constructeur	
`protected`	**`RecordedServiceType`**`(java.lang.String name)` Fournit une instance de RecordedServiceType.

Méthodes héritées de la classe javax.tv.service.ServiceType
`toString`

Méthodes héritées de la classe java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Détail du champ

`RECORDED_SERVICE_TYPE`

`public static final javax.tv.service.ServiceType` **`RECORDED_SERVICE_TYPE`**

ServiceType pour un Recorded service.

Détail pour le constructeur

RecordedServiceType

```
protected RecordedServiceType(java.lang.String name)
```

Fournit une instance de RecordedServiceType.

Paramètres:

`name` – Le nom de chaîne de ce type (c'est-à-dire "RECORDED_SERVICE").

D.1.10 Classe RecordingChangedEvent

```
org.ocap.shared.dvr
        Class RecordingChangedEvent

java.lang.Object
  +--java.util.EventObject
       +--org.ocap.shared.dvr.RecordingChangedEvent
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **RecordingChangedEvent**

étend java.util.EventObject

Evénement utilisé pour notifier aux auditeurs les changements de la liste des demandes d'enregistrement conservées par le RecordingManager.

Voir aussi:

Formes en série

Résumé du champ

`static int`	**`ENTRY_ADDED`** Une nouvelle `RecordingRequest` a été ajoutée.
`static int`	**`ENTRY_DELETED`** Une `RecordingRequest` a été supprimée.
`static int`	**`ENTRY_STATE_CHANGED`** L'état d'une `RecordingRequest` a changé.

Champs hérités d'une classe java.util.EventObject

`source`

Résumé pour le constructeur

`RecordingChangedEvent``(RecordingRequest source, int newState, int oldState)`
Construit l'événement.

Résumé pour la méthode	
int	**getChange**() Retourne le changement à la RecordingRequest.
int	**getOldState**() Retourne l'ancien état pour la RecordingRequest.
RecordingRequest	**getRecordingRequest**() Retourne la RecordingRequest qui a causé l'événement.
int	**getState**() Retourne le nouvel état pour la RecordingRequest.

Méthodes héritées de la classe java.util.EventObject
getSource, toString

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail du champ

ENTRY_ADDED

```
public static final int ENTRY_ADDED
```

Une nouvelle RecordingRequest a été ajoutée.

Voir aussi:

Valeurs de champ constantes

ENTRY_DELETED

```
public static final int ENTRY_DELETED
```

une RecordingRequest a été supprimée.

Voir aussi:

Valeurs de champ constantes

ENTRY_STATE_CHANGED

```
public static final int ENTRY_STATE_CHANGED
```

L'état d'une RecordingRequest a changé

Voir aussi:

Valeurs de champ constantes

Détail pour le constructeur

RecordingChangedEvent

```
public RecordingChangedEvent(RecordingRequest source,
                             int newState,
                             int oldState)
```

Construit l'événement.

Paramètres:

`source` – la `RecordingRequest` qui a causé l'événement.

`newState` – l'état dans lequel se trouve maintenant la `RecordingRequest`.

`oldState` – l'état dans lequel se trouvait la `RecordingRequest` avant le changement d'état.

Détail pour la méthode

getRecordingRequest

```
public RecordingRequest getRecordingRequest()
```

Retourne la `RecordingRequest` qui a causé l'événement.

Retourne:

la `RecordingRequest` qui a causé l'événement.

---

getChange

```
public int getChange()
```

Retourne le changement de la `RecordingRequest`.

Retourne:

si l'entrée a été ajoutée, supprimée ou modifiée.

---

getState

```
public int getState()
```

Retourne le nouvel état pour la `RecordingRequest`.

Retourne:

le nouvel état.

---

getOldState

```
public int getOldState()
```

Retourne l'ancien état de la demande `RecordingRequest`.

Retourne:

l'ancien état.

D.1.11 Interface RecordingChangedListener

```
org.ocap.shared.dvr
       Interface RecordingChangedListener
```

Toutes les super interfaces:

java.util.EventListener

---

interface publique **RecordingChangedListener**

étend java.util.EventListener

L'auditeur reçoit les changements de la liste d'enregistrements entretenue par le RecordingManager.

Résumé pour la méthode	
void	**recordingChanged**(RecordingChangedEvent e) Notifie au RecordingChangedListener un événement généré par le RecordingManager.

Détail pour la méthode

recordingChanged

```
public void recordingChanged(RecordingChangedEvent e)
```

Notifie au RecordingChangedListener un événement généré par le RecordingManager. Les événements sont générés lorsqu'il y a des changements dans la liste des demandes d'enregistrement entretenue par le gestionnaire d'enregistrements.

Paramètres:

e – L'événement généré.

D.1.12 Classe RecordingFailedException

```
org.ocap.shared.dvr
       Class RecordingFailedException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.dvr.RecordingFailedException
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **RecordingFailedException**

étend java.lang.Exception

Cette exception est retournée lorsque les applications invoquent la getFailedException() pour un échec de demande d'enregistrement ou une demande d'enregistrement incomplète.

Voir aussi:

Forme en série

Résumé du champ	
static int	**ACCESS_WITHDRAWN** Code de cause: L'enregistrement ne s'est pas bien terminé parce que l'accès au service ou à certains de ses composants a été retiré par le système avant l'achèvement programmé de l'enregistrement.
static int	**CA_REFUSAL** Code de cause: L'enregistrement a échoué car le système d'accès conditionnel refuse de le permettre.
static int	**CONTENT_NOT_FOUND** Code de cause: L'enregistrement a échoué parce que le contenu demandé n'a pu être trouvé dans le réseau.
static int	**INSUFFICIENT_RESOURCES** Code de cause: L'enregistrement a échoué à cause du manque de ressources nécessaires pour présenter ce service.
static int	**OUT_OF_BANDWIDTH** Code de cause: L'enregistrement a échoué du fait du manque de bande passante IO pour enregistrer ce programme.
static int	**RESOLUTION_ERROR** Code de cause: La demande d'enregistrement a échoué à cause d'erreurs dans la résolution de demande.
static int	**RESOURCES_REMOVED** Code de cause: L'enregistrement ne s'est pas bien terminé parce que les ressources nécessaires pour présenter le service ont été retirées avant l'achèvement programmé de l'enregistrement.
static int	**SERVICE_VANISHED** Code de cause: L'enregistrement ne s'est pas bien terminé parce que le service a disparu du réseau avant l'achèvement de l'enregistrement.
static int	**SPACE_FULL** Code de cause: L'enregistrement ne s'est pas terminé faute d'espace mémoire.
static int	**TUNED_AWAY** Code de cause: L'enregistrement ne s'est pas bien terminé parce que l'application a choisi un autre service dans le contexte de service.
static int	**TUNING_FAILURE** Code de cause: L'enregistrement a échoué à cause de problèmes de réglage.
static int	**USER_STOP** Code de cause: L'application a terminé l'enregistrement en utilisant la méthode LeafRecordingRequest.stop ou en invoquant l'arrêt sur le contexte de service (si la spécification d'enregistrement est une instance de ServiceContextRecordingSpec).

Résumé pour le constructeur
RecordingFailedException() Construit une exception RecordingFailedException sans message de détail

Résumé pour la méthode	
int	**getReason**() Rapporte la raison pour laquelle la demande d'enregistrement a échoué à bien se terminer.

Méthodes héritées de la classe java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail du champ

CA_REFUSAL

public static final int **CA_REFUSAL**

Code de cause: L'enregistrement a échoué parce que le système CA refuse de le permettre.

Voir aussi:

Valeurs de champ constantes

CONTENT_NOT_FOUND

public static final int **CONTENT_NOT_FOUND**

Code de cause: L'enregistrement a échoué parce que le contenu demandé n'a pu être trouvé sur le réseau.

Voir aussi:
Valeurs de champ constantes

TUNING_FAILURE

public static final int **TUNING_FAILURE**

Code de cause: L'enregistrement a échoué pour des problèmes de réglage.

Voir aussi:

Valeurs de champ constantes

INSUFFICIENT_RESOURCES

```
public static final int INSUFFICIENT_RESOURCES
```

Code de cause: L'enregistrement a échoué à cause du manque de ressources nécessaires pour présenter ce service.

Voir aussi:

Valeurs de champ constantes

ACCESS_WITHDRAWN

```
public static final int ACCESS_WITHDRAWN
```

Code de cause: L'enregistrement ne s'est pas bien terminé parce que l'accès au service ou à un de ses composants a été retiré par le système avant l'achèvement programmé de l'enregistrement.

Voir aussi:

Valeurs de champ constantes

RESOURCES_REMOVED

```
public static final int RESOURCES_REMOVED
```

Code de cause: L'enregistrement ne s'est pas bien terminé parce que les ressources nécessaires pour présenter le service ont été retirées avant l'achèvement programmé de l'enregistrement.

Voir aussi:

Valeurs de champ constantes

SERVICE_VANISHED

```
public static final int SERVICE_VANISHED
```

Code de cause: L'enregistrement ne s'est pas bien terminé parce que le service a disparu du réseau avant l'achèvement de l'enregistrement.

Voir aussi:

Valeurs de champ constantes

TUNED_AWAY

```
public static final int TUNED_AWAY
```

Code de cause: L'enregistrement ne s'est pas bien terminé parce que l'application a choisi un autre service dans le contexte de service. Ceci n'est applicable que si la spécification d'enregistrement pour la demande d'enregistrement est une instance de ServiceContextRecordingSpec.

Voir aussi:

Valeurs de champ constantes

USER_STOP

```
public static final int USER_STOP
```

Code de cause: L'application a terminé l'enregistrement en utilisant la méthode LeafRecordingRequest.stop ou en invoquant l'arrêt sur le contexte de service (si la spécification d'enregistrement est une instance de ServiceContextRecordingSpec).

Voir aussi:

Valeurs de champ constantes

SPACE_FULL

```
public static final int SPACE_FULL
```

Code de cause: L'enregistrement ne pourra pas s'achever à cause d'un manque d'espace mémoire.

Voir aussi:

Valeurs de champ constantes

OUT_OF_BANDWIDTH

```
public static final int OUT_OF_BANDWIDTH
```

Code de cause: L'enregistrement a échoué à cause d'un manque de bande passante IO pour enregistrer ce programme.

Voir aussi:

Valeurs de champ constantes

RESOLUTION_ERROR

```
public static final int RESOLUTION_ERROR
```

Code de cause: La demande d'enregistrement a échoué à cause d'erreurs dans la résolution de la demande.

Voir aussi:

Valeurs de champ constantes

Détail pour le constructeur

RecordingFailedException

```
public RecordingFailedException()
```

Construit une exception RecordingFailedException sans message de détail.

Détail pour la méthode

getReason

```
public int getReason()
```

Rapporte la raison pour laquelle la demande d'enregistrement a échoué à s'achever avec succès.

Retourne:

le code de cause pour laquelle la demande d'enregistrement a échoué à s'achever avec succès.

D.1.13 Classe RecordingManager

```
org.ocap.shared.dvr
        Class RecordingManager

java.lang.Object
  +--org.ocap.shared.dvr.RecordingManager
```

---

classe abstraite publique **RecordingManager**

étend java.lang.Object

RecordingManager représente l'entité qui effectue les enregistrements.

Résumé pour le constructeur	
protected	**RecordingManager**() Constructeur pour les instances de cette classe.

Résumé pour la méthode	
abstract void	**addRecordingChangedListener**(RecordingChangedListener rcl) Ajoute un surveillant d'événement des changements d'état des demandes d'enregistrement.
abstract RecordingList	**getEntries**() Obtient la liste des entrées conservées par le RecordingManager.
abstract RecordingList	**getEntries**(RecordingListFilter filter) Obtient la liste des demandes d'enregistrement satisfaisant au filtre spécifié.
static RecordingManager	**getInstance**() Obtient la seule instance de RecordingManager.
abstract RecordingRequest	**getRecordingRequest**(int id) Recherche une demande d'enregistrement d'après l'identifiant.

abstract RecordingRequest	**record**(RecordingSpec source) Enregistre le ou les flux conformément au paramètre de source.
abstract void	**removeRecordingChangedListener**(RecordingchangedListener rc1) Retire un surveillant d'événement enregistré de changements des états des demandes d'enregistrement.

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Détail pour le constructeur

RecordingManager

protected **RecordingManager**()

Constructeur pour les instances de cette classe. Ce constructeur est fourni pour l'utilisation des implémentations et spécifications qui étendent cette spécification. Les applications ne doivent pas définir de sous-classes de cette classe. Les implémentations ne sont pas obligées de se comporter correctement si de telles sous-classes définies par l'application sont utilisées.

Détail pour la méthode

getEntries

public abstract RecordingList **getEntries**()

Obtient la liste des entrées conservées par le RecordingManager. Cette liste comporte à la fois les demandes d'enregistrement parentes et les finales. Pour les applications avec RecordingPermission("read", "own"), seules seront retournées les demandes RecordingRequests dont l'application appelante a une visibilité comme définie par des attributs de sécurité spécifiques d'une RecordingRequest quelconque. Pour les applications avec RecordingPermission("read", "*"), toutes les RecordingRequests seront retournées.

Retourne:

une instance de RecordingList

Invocation:

java.lang.SecurityException – si l'application appelante n'a pas de RecordingPermission("read",..) ou de RecordingPermission("*",..)

---

getEntries

```
public abstract RecordingList getEntries(RecordingListFilter filter)
```

Obtient la liste des demandes d'enregistrement qui correspondent au filtre spécifié. Pour les applications avec RecordingPermission("read", "own"), seules seront retournées les RecordingRequests dont l'application appelante a la visibilité définie des attributs de sécurité spécifiques d'une RecordingRequest quelconque. Pour les applications avec RecordingPermission("read", "*"), toutes les RecordingRequests qui satisfont au filtre spécifié seront retournées.

Paramètres:

`filter` – le filtre à utiliser sur l'ensemble total des demandes d'enregistrement.

Retourne:

une instance de RecordingList

Invocation:

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("read",..) ou de RecordingPermission("*",..)

---

addRecordingChangedListener

```
public abstract void addRecordingChangedListener(RecordingChangedListener rcl)
```

Ajoute un surveillant d'événement des changements d'état des demandes d'enregistrement. Pour les applications avec RecordingPermission("read", "own"), le paramètre de surveillant sera seulement informé des changements qui affectent les demandes RecordingRequest dont l'application appelante a la visibilité définie par tout attribut de sécurité spécifique de RecordingRequest. Pour les applications avec RecordingPermission("read", "*"), le paramètre listener sera informé de tous les changements.

Paramètres:

`rcl` – Le surveillant *(listener)* à enregistrer.

Invocation:

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("read",..) ou de RecordingPermission("*",..)

---

removeRecordingChangedListener

```
public abstract void
removeRecordingChangedListener(RecordingChangedListener rcl)
```

Retire un surveillant d'événement enregistré de changements d'état de demandes d'enregistrement. Si le surveillant spécifié n'est pas enregistré, cette méthode n'a pas d'effet.

Paramètres:

`rcl` – le surveillant à retirer.

---

record

```
public abstract RecordingRequest record(RecordingSpec source)
                                 throws java.lang.IllegalArgumentException,
                                        AccessDeniedException
```

Enregistre le ou les flux conformément au paramètre de source. La sous-classe concrète de RecordingSpec peut définir une sémantique supplémentaire à appliquer lorsque des instances de cette sous-classe sont utilisées.

Paramètres:

`source` – spécification du ou des flux à enregistrer et de la façon dont ils sont à enregistrer.

Retourne:

une instance de RecordingRequest qui représente l'enregistrement ajouté.

Invocation:

`java.lang.IllegalArgumentException` – si la source est une classe définie par l'application ou est définie dans la sous-classe concrète de RecordingSpec pour les instances de cette classe.

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("create",..) ou de RecordingPermission("*",..)

---

getInstance

```
public static RecordingManager getInstance()
```

Obtient la seule instance de RecordingManager.

Retourne:

une instance de RecordingManager

---

getRecordingRequest

```
public abstract RecordingRequest getRecordingRequest(int id)
                                 throws java.lang.IllegalArgumentException
```

Recherche une demande d'enregistrement à partir de l'identifiant. Les implémentations de cette méthode devraient être optimisées en considérant la vraisemblance d'un très grand nombre de demandes d'enregistrement. Pour les applications avec RecordingPermission("read", "own"), seules seront retournées les RecordingRequests dont l'application appelante a une visibilité comme définie par des attributs quelconques de sécurité spécifiques de la RecordingRequest.

Paramètres:

`id` – un identifiant tel que retourné par RecordingRequest.getId

Retourne:

La RecordingRequest correspondante

Invocation:

`java.lang.IllegalArgumentException` – si il n'y a pas de demande d'enregistrement correspondant à cet identifiant ou si la demande d'enregistrement n'est pas visible comme défini avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("read",..) ou de RecordingPermission("*",..)

Voir aussi:

`RecordingRequest.getId()`

D.1.14 Classe RecordingPermission

org.ocap.shared.dvr

Class RecordingPermission

```
java.lang.Object
  +--java.security.Permission
       +--org.ocap.shared.dvr.RecordingPermission
```

Toutes les interfaces implémentées:

java.security.Guard, java.io.Serializable

---

classe finale publique **RecordingPermission**

étend java.security.Permission

Contrôle l'accès aux caractéristiques d'enregistrement par une application. Le nom peut être une des valeurs de la liste suivante:

– "create" – programme une RecordingRequest
– "read" – obtient la liste des RecordingRequests
– "modify" – modifie les propriétés ou données spécifiques d'application pour une RecordingRequest
– "delete" – supprime une RecordingRequest, y compris le contenu enregistré
– "cancel" – annule une RecordingRequest en instance
– "*" – toutes les valeurs ci-dessus.

L'action peut être "own" et "*". L'action "own" est destinée à être utilisée par des applications normales. L'action "*" est destinée à être utilisée seulement par des applications à privilège particulier et permet à l'opération définie par le nom d'être appliquée à toutes les RecordingRequests indépendamment de toute restriction spécifique de l'application associée à la RecordingRequest.

Accorder cette permission doit inclure d'accorder l'accès à tout appareil de mémorisation nécessaire pour les opérations spécifiées dans le paramètre de nom. Aucune permission supplémentaire de niveau inférieur (par exemple, FilePermission) n'est ensuite nécessaire.

Voir aussi:

Forme en série

Résumé pour le constructeur
`RecordingPermission``(java.lang.String name, java.lang.String action)` Crée une nouvelle RecordingPermission avec le nom et l'action spécifiés.

Résumé pour la méthode	
boolean	**equals**(java.lang.Object obj) Vérifie l'égalité de deux objets RecordingPermission
java.lang.String	**getActions**() Retourne les actions telles que passées dans le constructeur.
int	**hashCode**() Retourne la valeur de code de hachage pour cet objet.
boolean	**implies**(java.security.Permission p) Vérifie si cette RecordingPermission "implique" la permission spécifiée.

Méthodes héritées de la classe java.security.Permission
checkGuard, getName, newPermissionCollection, toString

Méthodes héritées de la classe java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Détail pour le constructeur

RecordingPermission

```
public RecordingPermission(java.lang.String name,
                           java.lang.String action)
```

Crée une nouvelle RecordingPermission avec le nom et l'action spécifiés.

Paramètres:

`name` – "create", "read", "modify", "delete", "cancel" ou "*"

`action` – "own" ou "*"

Détail pour la méthode

implies

```
public boolean implies(java.security.Permission p)
```

Vérifie si cette RecordingPermission "implique" la permission spécifiée.

Paramètres:

`p` – permission de vérifier

Retourne:

vrai si la permission spécifiée est impliquée par cet objet, faux sinon.

equals

```
public boolean equals(java.lang.Object obj)
```

Vérifie l'égalité de deux objets RecordingPermission.

Paramètres:

`obj` – l'objet dont l'égalité est testée avec cet objet.

Retourne:

vrai si obj est une RecordingPermission avec les mêmes nom et action que dans cet objet RecordingPermission.

---

hashCode

```
public int hashCode()
```

Retourne la valeur de code de hachage pour cet objet.

Retourne:

une valeur de code de hachage pour cet objet.

---

getActions

```
public java.lang.String getActions()
```

Retourne les actions telles que passées dans le constructeur.

Retourne:

les actions sous forme d'une chaîne.

D.1.15 Classe RecordingProperties

```
org.ocap.shared.dvr
        Class RecordingProperties

java.lang.Object
  +--org.ocap.shared.dvr.RecordingProperties
```

---

classe abstraite publique **RecordingProperties**

étend java.lang.Object

Classe de base pour les propriétés de spécification qui définissent comment faire un enregistrement.

Résumé pour le constructeur
RecordingProperties(long expirationPeriod) Constructeur

Résumé pour la méthode	
long	**getExpirationPeriod**() Retourne la valeur de la période d'expiration.

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Détail pour le constructeur

RecordingProperties

public **RecordingProperties**(long expirationPeriod)

Constructeur

Paramètres:

expirationPeriod – La période en secondes après l'initialisation de l'enregistrement lorsqu'une demande d'enregistrement finale avec ses propriétés d'enregistrement est réputée être arrivée à expiration.

Détail pour la méthode

getExpirationPeriod

public long **getExpirationPeriod**()

Retourne la valeur de la période d'expiration.

Retourne:

La période d'expiration telle que passée dans le constructeur.

D.1.16 Interface RecordingRequest

org.ocap.shared.dvr
Interface RecordingRequest

Toutes les sous-interfaces connues:

LeafRecordingRequest, ParentRecordingRequest

---

interface publique **RecordingRequest**

Cette interface représente les informations correspondant à une demande d'enregistrement. La demande d'enregistrement représentée par cette interface peut correspondre à une seule demande d'enregistrement ou à une série d'autres demandes d'enregistrement. Les demandes d'enregistrement sont hiérarchiques par nature. Les implémentations peuvent résoudre une demande d'enregistrement en une seule demande d'enregistrement ou en une série d'autres demandes d'enregistrement dont chacune peut être ultérieurement résolue en une seule demande d'enregistrement ou en une série de demandes d'enregistrement. Par exemple, une demande d'enregistrement pour "le sexe et la ville" peut se résoudre en plusieurs demandes d'enregistrement, chacune pour une session particulière de

la série. Chacune de ces demandes d'enregistrement peut se résoudre ensuite en plusieurs demandes d'enregistrement, chacune pour un seul épisode. L'implémentation crée une demande d'enregistrement en réponse à la méthode RecordingManager.record(RecordingSpec). L'implémentation crée aussi des demandes d'enregistrement lorsqu'une demande d'enregistrement est plus détaillée.

Une demande d'enregistrement peut être une demande d'enregistrement parente ou une demande d'enregistrement finale. Les états d'une demande d'enregistrement sont définis dans `ParentRecordingRequest` et `LeafRecordingRequest`. Une demande d'enregistrement peut être dans l'un quelconque des états correspondant à une demande d'enregistrement finale ou à une demande d'enregistrement parente.

Résumé pour la méthode	
`void`	**`addAppData`**`(java.lang.String key, java.io.Serializable data)` Ajoute des données privées spécifiques de l'application.
`void`	**`delete`**`()` Supprime la demande d'enregistrement de la base de données.
`java.io.Serializable`	**`getAppData`**`(java.lang.String key)` Obtient les données d'application correspondant à la clé spécifiée.
`org.dvb.application.AppID`	**`getAppID`**`()` Obtient l'identifiant d'application de l'application qui possède cette demande d'enregistrement.
`int`	**`getId`**`()` Retourne un identifiant pour cette demande d'enregistrement.
`java.lang.String[]`	**`getKeys`**`()` Obtient toutes les données spécifiques d'application associées à cette demande d'enregistrement.
`RecordingRequest`	**`getParent`**`()` Obtient la demande d'enregistrement parente correspondant à cette demande d'enregistrement.
`RecordingSpec`	**`getRecordingSpec`**`()` Retourne la RecordingSpec correspondant à la demande d'enregistrement.
`RecordingRequest`	**`getRoot`**`()` Obtient la demande d'enregistrement racine correspondant à cette demande d'enregistrement.
`int`	**`getState`**`()` Retourne l'état de la demande d'enregistrement.
`boolean`	**`isRoot`**`()` Vérifie si la demande d'enregistrement était une demande d'enregistrement racine générée lorsque l'application a invoqué le RecordingManager.record(..)

void	**removeAppData**(java.lang.String key) Supprime les données privées spécifiques de l'application correspondant à la clé spécifiée.
void	**reschedule**(RecordingSpec newRecordingSpec) Modifie les détails d'une demande d'enregistrement.
void	**setRecordingProperties**(RecordingProperties properties) Modifie les RecordingProperties correspondant à la RecordingSpec pour cette demande d'enregistrement.

Détail pour la méthode

getState

```
public int getState()
```

Retourne l'état de la demande d'enregistrement.

Retourne:

Etat de la demande d'enregistrement.

---

isRoot

```
public boolean isRoot()
```

Vérifie si la demande d'enregistrement était une demande d'enregistrement racine générée lorsque l'application a invoqué la méthode RecordingManager.record(..). L'implémentation devrait créer une demande d'enregistrement racine correspondant à chaque invocation réussie de la méthode d'enregistrement.

Retourne:

vrai, si la demande d'enregistrement est une demande d'enregistrement racine, faux si la demande d'enregistrement a été générée durant le processus de résolution d'une autre demande d'enregistrement.

---

getRoot

```
public RecordingRequest getRoot()
```

Obtient la demande d'enregistrement racine correspondant à cette demande d'enregistrement. Une demande d'enregistrement racine est la demande d'enregistrement qui a été retournée lorsque l'application a invoqué la méthode RecordingManager.record(..).

Si la demande d'enregistrement en cours est une demande d'enregistrement racine, la demande d'enregistrement en cours est retournée.

Retourne:

la demande d'enregistrement racine pour cette demande d'enregistrement, nul si l'application n'a pas de permission d'accès en lecture à la demande d'enregistrement racine.

---

getParent

```
public RecordingRequest getParent()
```

Obtient la demande d'enregistrement parente correspondant à cette demande d'enregistrement.

Retourne:

la demande d'enregistrement parente pour cette demande d'enregistrement, nul si l'application n'a pas de permission d'accès en lecture pour la demande d'enregistrement parente ou si cette demande d'enregistrement est la demande d'enregistrement racine.

---

getRecordingSpec

```
public RecordingSpec getRecordingSpec()
```

Retourne la RecordingSpec correspondant à la demande d'enregistrement. Cela sera soit la source comme spécifié dans l'invocation à la méthode record(..) qui a causé la création de cette demande d'enregistrement, soit la RecordingSpec générée par le système durant la résolution de la RecordingSpec spécifiée par l'application d'origine. Toute modification de la RecordingSpec due à des invocations ultérieures aux méthodes de SetRecordingProperties sur cette instance sera reflétée sur la RecordingSpec retournée.

Lorsque l'implémentation génère une demande d'enregistrement pendant la résolution d'une autre demande d'enregistrement, une nouvelle instance de RecordingSpec est créée avec une copie identique de la RecordingProperties de la demande d'enregistrement parente.

Retourne:

une RecordingSpec contenant des informations sur cette demande d'enregistrement.

---

setRecordingProperties

```
public void setRecordingProperties(RecordingProperties properties)
                            throws java.lang.IllegalStateException,
                                   AccessDeniedException
```

Modifie les RecordingProperties correspondant à la RecordingSpec pour cette demande d'enregistrement. Les applications peuvent changer toute propriété associée à une demande d'enregistrement en invoquant cette méthode. Changer les propriétés peut avoir pour résultat des changements des états de cette demande d'enregistrement. Changer les propriétés d'une demande d'enregistrement parente ne changera pas automatiquement les propriétés des demandes d'enregistrement filles qui sont déjà créées. Toute demande d'enregistrement fille créée après l'invocation de cette méthode héritera des nouvelles valeurs des propriétés.

Paramètres:

`properties` – les nouvelles propriétés d'enregistrement à établir.

Invocation:

`java.lang.IllegalStateException` – si le changement d'un des paramètres qui a été modifié dans les nouvelles propriétés d'enregistrement n'est pas légal pour l'état en cours de la demande d'enregistrement.

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("modify",..) ou de RecordingPermission("*",..)

---

delete

```
public void delete()
            throws AccessDeniedException
```

Supprime la demande d'enregistrement de la base de données. La méthode supprime la demande d'enregistrement, toutes ses demandes d'enregistrement descendantes, ainsi que les objets `RecordedService` correspondants et tous les flux élémentaires enregistrés (par exemple, fichiers et entrées de répertoire) associés au RecordedService. Si une application invoque une méthode quelconque sur les références périmées d'objets supprimés, l'implémentation doit lancer une IllegalStateException.

Si la demande d'enregistrement est dans l'état IN_PROGRESS, l'implémentation va arrêter l'enregistrement avant de supprimer la demande d'enregistrement. Si un RecordedService a été présenté lorsqu'elle a été supprimée, un événement `PresentationTerminatedEvent` sera envoyé avec la cause SERVICE_VANISHED *(service disparu)*.

Invocation:

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("delete",..) ou de RecordingPermission("*",..)

---

addAppData

```
public void addAppData(java.lang.String key,
                       java.io.Serializable data)
                throws NoMoreDataEntriesException,
                       AccessDeniedException
```

Ajoute des données privées spécifiques d'application. Si la clé est déjà utilisée, les données correspondant à la clé sont recouvertes.

Paramètres:

`key` – l'identifiant sous lequel les données sont à ajouter

`data` – les données à ajouter

Invocation:

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("modify",..) ou de RecordingPermission("*",..)

`java.lang.IllegalArgumentException` – si la taille des données est supérieure à la taille acceptée par l'implémentation dans les contraintes de la spécification de GEM pour l'enregistrement.

`NoMoreDataEntriesException` – si la mémorisation de ces données excède le nombre des entrées acceptées par l'implémentation dans la limite des contraintes de la spécification de GEM pour l'enregistrement.

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

getAppID

```
public org.dvb.application.AppID getAppID()
```

Obtient l'identifiant d'application de l'application qui possède cette demande d'enregistrement. Le propriétaire d'une demande d'enregistrement racine est l'application qui invoque la méthode RecordingManager.record(..). Le propriétaire d'une demande d'enregistrement non racine est le propriétaire de la demande d'enregistrement racine.

Retourne:

Identifiant d'application de l'application propriétaire.

---

getKeys

```
public java.lang.String[] getKeys()
```

Obtient toutes les données spécifiques d'application associées à cette demande d'enregistrement.

Retourne:

Toutes les clés correspondant à la RecordingRequest; nul s'il n'y a pas de données d'application.

---

getAppData

```
public java.io.Serializable getAppData(java.lang.String key)
```

Obtient les données d'application correspondant à la clé spécifiée.

Paramètres:

`key` – la clé sous laquelle toutes données sont à retourner.

Retourne:

les données d'application correspondant à la clé spécifiée; nul s'il n'y a pas de données correspondant à la clé spécifiée.

---

removeAppData

```
public void removeAppData(java.lang.String key)
                   throws AccessDeniedException
```

Supprime les données privées spécifiques d'application correspondant à la clé spécifiée. Cette méthode s'efface en silence s'il n'y a pas de données correspondant à la clé.

Paramètres:

`key` – clé sous laquelle les données sont à supprimer.

Invocation:

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("modify",..) ou de RecordingPermission("*",..)

reschedule

```
public void reschedule(RecordingSpec newRecordingSpec)
                throws AccessDeniedException
```

Modifie les détails d'une demande d'enregistrement. La demande d'enregistrement doit être réévaluée sur la base de la RecordingSpec nouvellement fournie. La reprogrammation d'une demande d'enregistrement racine peut avoir pour résultat des transitions d'état pour la demande d'enregistrement racine ou ses demandes d'enregistrement filles. La reprogrammation d'une demande d'enregistrement racine peut aussi avoir pour résultat la programmation d'une ou plusieurs demandes d'enregistrement filles, ou la suppression d'une ou plusieurs demandes d'enregistrement filles en instance.

NOTE – Si la demande d'enregistrement ou une de ses demandes d'enregistrement filles est dans l'état IN_PROGRESS_STATE ou IN_PROGRESS_INSUFFICIENT_SPACE_STATE, tout changement de l'heure de début doit être ignoré. Dans ce cas, tous les autres paramètres valides s'appliquent. Si la nouvelle valeur d'un paramètre n'est pas valide (par exemple, l'heure de début et la durée sont passées), l'implémentation doit ignorer ce paramètre. Les enregistrements en cours doivent continuer sans interruption, si la nouvelle spécification d'enregistrement n'exige pas que l'enregistrement soit arrêté.

Paramètres:

`newRecordingSpec` – nouvelle spécification d'enregistrement qui doit être utilisée pour reprogrammer la RecordingRequest racine.

Invocation:

`java.lang.IllegalArgumentException` – si la spécification nouvellement enregistrée et la spécification d'enregistrement en cours pour la demande d'enregistrement sont dans des sous-classes différentes de RecordingSpec.

`AccessDeniedException` – si l'application appelante n'est pas autorisée à effectuer cette opération avec des attributs de sécurité spécifiques de la demande RecordingRequest.

`java.lang.SecurityException` – si l'application appelante n'a pas de RecordingPermission("modify",..) ou de RecordingPermission("*",..)*

---

getId

```
public int getId()
```

Retourne un identifiant pour cette demande d'enregistrement. L'identifiant doit identifier de façon univoque cette demande d'enregistrement parmi toutes les autres dans le terminal d'enregistrement GEM. L'identifiant doit être associé de façon permanente à cette demande d'enregistrement tant que cette demande d'enregistrement reste dans le terminal d'enregistrement GEM et en particulier, elle doit survivre aux interruptions d'alimentation électrique du terminal d'enregistrement GEM. Ceci est destiné à permettre aux applications de mémoriser ces identifiants dans des mémoires persistantes pour les restituer ultérieurement à d'autres applications ou autres instances de la même application.

Comme les identifiants peuvent être conservés dans des mémoires persistantes par les applications, les implémentations ne devraient pas réutiliser les identifiants des demandes d'enregistrement qui ne sont plus détenues dans le terminal d'enregistrement GEM, car cela pourrait créer une confusion pour les applications qui ont toujours les références de ces demandes d'enregistrement dans leurs mémoires persistantes.

Retourne:

un identifiant

Voir aussi:

`RecordingManager.getRecordingRequest(int)`

D.1.17 Classe RecordingSpec

```
org.ocap.shared.dvr
        Class RecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
```

Sous-classes directement connues:

LocatorRecordingSpec, ServiceContextRecordingSpec, ServiceRecordingSpec

---

classe abstraite publique **RecordingSpec**

étend java.lang.Object

Classe de base pour spécifier ce qu'il faut enregistrer et comment l'enregistrer.

Résumé pour le constructeur
`RecordingSpec``(RecordingProperties properties)` Constructeur.

Résumé pour la méthode	
`RecordingProperties`	**`getProperties`**`()` Retourne la description de la façon de faire cet enregistrement.

Méthodes héritées de la classe java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Détail pour le constructeur

RecordingSpec

`public` **`RecordingSpec`**`(RecordingProperties properties)`

Constructeur.

Paramètres:

`properties` – Définition de la façon de faire l'enregistrement.

Détail pour la méthode

getProperties

```
public RecordingProperties getProperties()
```

Retourne la description de la façon de faire l'enregistrement.

Retourne:

les propriétés à utiliser pour l'enregistrement.

D.1.18 Classe RecordingTerminatedEvent

```
org.ocap.shared.dvr
        Class RecordingTerminatedEvent

java.lang.Object
  +--java.util.EventObject
      +--javax.tv.service.selection.ServiceContextEvent
          +--org.ocap.shared.dvr.RecordingTerminatedEvent
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **RecordingTerminatedEvent**

étend javax.tv.service.selection.ServiceContextEvent

Un événement qui notifie que l'enregistrement est terminé pour le `ServiceContext`. Cet événement est généré par un `ServiceContext` qui présente un service en différé ou un service en cours d'enregistrement. La présentation n'est pas terminée lorsque le point de lecture est différé. Cet événement n'est généré que lorsque le point de lecture n'est pas le même que le point de direct. Un événement `PresentationTerminatedEvent` sera généré lorsque le point de lecture rattrape le point de terminaison d'enregistrement.

Voir aussi:

Forme en série

Résumé du champ	
`static int`	**`ACCESS_WITHDRAWN`** Code de cause: L'accès au service ou certains de ses composants a été supprimé par le système.
`static int`	**`RESOURCES_REMOVED`** Code de cause: Les ressources nécessaires à l'enregistrement du service ont été retirées.
`static int`	**`SCHEDULED_STOP`** Code de cause: L'enregistrement s'est terminé normalement comme programmé.
`static int`	**`SERVICE_VANISHED`** Code de cause: Le service a disparu du réseau.

static int	**USER_STOP** Code de cause: L'utilisateur a demandé l'arrêt de l'enregistrement.

Champs hérités de la classe java.util.EventObject

source

Résumé pour le constructeur

RecordingTerminatedEvent(javax.tv.service.selection.ServiceContext source, int reason)
Construit l'événement.

Résumé pour la méthode

int	**getReason**() Retourne la cause de la fin de l'enregistrement.

Méthodes héritées de la classe javax.tv.service.selection.ServiceContextEvent

getServiceContext

Méthodes héritées de la classe java.util.EventObject

getSource, toString

Méthodes héritées de la classe java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail du champ

SERVICE_VANISHED

public static final int **SERVICE_VANISHED**

Code de cause: Le service a disparu du réseau.

Voir aussi:

Valeurs de champ constantes

RESOURCES_REMOVED

```
public static final int RESOURCES_REMOVED
```

Code de cause: Les ressources nécessaires pour enregistrer le service ont été retirées. Sera généré si la méthode d'arrêt `ServiceContext` est invoquée.

Voir aussi:

Valeurs de champ constantes

ACCESS_WITHDRAWN

```
public static final int ACCESS_WITHDRAWN
```

Code de cause: L'accès au service ou à certains de ses composants a été retiré par le système. La fin d'une période de visualisation gratuite d'un contenu d'IPPV en est un exemple.

Voir aussi:

Valeurs de champ constantes

SCHEDULED_STOP

```
public static final int SCHEDULED_STOP
```

Code de cause: L'enregistrement s'est terminé normalement comme programmé.

Voir aussi:

Valeurs de champ constantes

USER_STOP

```
public static final int USER_STOP
```

Code de cause: L'utilisateur a demandé l'arrêt de l'enregistrement. Mais aussi, si la méthode d'arrêt `RecordingRequest` est invoquée.

Voir aussi:

Valeurs de champ constantes

Détail pour le constructeur

RecordingTerminatedEvent

```
public
RecordingTerminatedEvent(javax.tv.service.selection.ServiceContext source,
                         int reason)
```

Construit l'événement.

Paramètre:

`source` – Le `ServiceContext` qui a généré l'événement.

Détail pour la méthode

getReason

```
public int getReason()
```

Retourne la cause de la fin de l'enregistrement.

Retourne:

Cause de la fin; voir les constantes dans cette classe.

D.1.19 Classe ServiceContextRecordingSpec

```
org.ocap.shared.dvr
        Class ServiceContextRecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
       +--org.ocap.shared.dvr.ServiceContextRecordingSpec
```

---

classe publique **ServiceContextRecordingSpec**

étend RecordingSpec

Spécifie une demande d'enregistrement sous la forme de ce qui est présenté sur un ServiceContext. Les flux qui sont présentés dans le paramètre ServiceContext indiqué sont enregistrés. Si le service d'où l'enregistrement est en cours est éteint, l'enregistrement DOIT s'arrêter. Si l'heure de début startTime est passée et si la source javax.tv.service.selection.ServiceContext est associée à une mémoire tampon de différé, les contenus de la mémoire tampon de différé peuvent être immédiatement mémorisés à la destination qui commence au startTime, si possible, jusqu'au point de diffusion en direct. Si la mémoire tampon de différé ne contient pas la source commençant au startTime, on enregistre autant que faire se peut de la source. Si le startTime est dépassé, mais qu'une mémoire tampon de différé ne peut pas être associée à l'enregistrement, l'enregistrement commence à partir du point de diffusion en direct. À partir de là, les contenus de la diffusion en direct sont enregistrés jusqu'à concurrence de la durée restante.

Lorsque des instances de cette classe sont passées à RecordingManager.record(..), les modes d'échec supplémentaires suivants doivent être appliqués:

– IllegalArgumentException DOIT être lancé si le serviceContext ne présente pas de service de diffusion ou si l'heure de début startTime est dans le futur.

– SecurityException DOIT être lancé si l'application n'a pas la permission d'accéder au contexte de service.

Lorsqu'une instance de cette spécification d'enregistrement est entrée comme paramètre à la méthode RecordingRequest.reschedule(..), une exception IllegalArgumentException est lancée si le paramètre de contexte de service est différent du contexte de service spécifié dans la spécification d'enregistrement en cours pour la demande d'enregistrement.

Résumé pour le constructeur

```
ServiceContextRecordingSpec
(javax.tv.service.selection.ServiceContext serviceContext,
java.util.Date startTime, long duration, RecordingProperties properties)
```

Constructeur.

Résumé pour la méthode	
long	**getDuration**() Retourne la durée passée comme argument au constructeur.
javax.tv.service.selection.ServiceContext	**getServiceContext**() Retourne le ServiceContext d'où enregistrer
java.util.Date	**getStartTime**() Retourne l'heure de début passée comme argument au constructeur.

Méthodes héritées de la classe org.ocap.shared.dvr.RecordingSpec
getProperties

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Détail pour le constructeur

ServiceContextRecordingSpec

```
public
ServiceContextRecordingSpec(javax.tv.service.selection.ServiceContext serviceCon
text,
                                    java.util.Date startTime,
                                    long duration,
                                    RecordingProperties properties)
                             throws java.lang.IllegalArgumentException
```

Constructeur

Paramètres:

`serviceContext` – ServiceContext d'où effectuer l'enregistrement.

`startTime` – Heure de début de l'enregistrement. Si l'heure de début est dans le futur lorsque la méthode RecordingManager.record(..) est invoquée avec cette ServiceContextRecordingSpec comme argument, la méthode d'enregistrement lancera une exception IllegalArgumentException.

`duration` – Durée à enregistrer en millisecondes.

`properties` – Définition de la façon dont l'enregistrement doit être effectué.

Invocation:

`java.lang.IllegalArgumentException` – si la durée est négative

Détail pour la méthode

getServiceContext

```
public javax.tv.service.selection.ServiceContext getServiceContext()
```

Retourne le ServiceContext d'où effectuer l'enregistrement.

Retourne:

l'instance de ServiceContext passée au constructeur.

---

getStartTime

```
public java.util.Date getStartTime()
```

Retourne l'heure de début passée comme argument au constructeur.

Retourne:

l'heure de début passée au constructeur.

---

getDuration

```
public long getDuration()
```

Retourne la durée passée comme argument au constructeur.

Retourne:

la durée passée au constructeur.

D.2 Paquetage de navigation d'enregistrement vidéo numérique partagé

```
org.ocap.shared.dvr.navigation
        Package
```


D.2.1 Classe RecordingStateFilter

```
org.ocap.shared.dvr.navigation
        Class RecordingStateFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
       +--org.ocap.shared.dvr.navigation.RecordingStateFilter
```

---

classe publique **RecordingStateFilter**

étend RecordingListFilter

Filtre à filtre sur la base des valeurs retournées par la méthode getState dans `RecordingRequest`.

Résumé pour le constructeur
`RecordingStateFilter``(int state)` Construit le filtre sur la base d'un type d'état particulier (PENDING, FAILED, etc.).

Résumé pour la méthode	
`boolean`	**`accept`**`(RecordingRequest entry)` Vérifie si la `RecordingRequest` donnée passe le filtre.
`int`	**`getFilterValue`**`()` Rapporte la valeur de l'état utilisé pour créer ce filtre.

Méthodes héritées de la classe org.ocap.shared.dvr.navigation.RecordingListFilter
`setCascadingFilter`

Méthodes héritées de la classe java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Détail pour le constructeur

RecordingStateFilter

`public` **`RecordingStateFilter`**`(int state)`

Construit le filtre sur la base d'un type d'état particulier (PENDING, FAILED, etc.).

Paramètres:

`state` – Valeur pour la correspondance d'état d'une instance de `RecordingRequest`.

Détail pour la méthode

getFilterValue

`public int` **`getFilterValue`**`()`

Rapporte la valeur de l'état utilisé pour créer ce filtre.

Retourne:

La valeur de l'état utilisé pour créer ce filtre.

accept

`public boolean` **`accept`**`(RecordingRequest entry)`

Vérifie si la `RecordingRequest` donnée passe le filtre.

Spécifié par:

`accept` dans la classe `RecordingListFilter`

Paramètres:

`entry` – Une RecordingRequest individuelle à évaluer par rapport à l'algorithme de filtrage.

Retourne:

`vrai` si la `RecordingRequest` contenue dans le paramètre RecordingRequest est dans l'état indiqué par la valeur du filtre; `faux` autrement.

D.2.2 Classe AppIDFilter

```
org.ocap.shared.dvr.navigation
       Class AppIDFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
      +--org.ocap.shared.dvr.navigation.AppIDFilter
```

---

classe publique **AppIDFilter**

étend RecordingListFilter

Filtre à filtre sur la base de AppID.

Résumé pour le constructeur
`AppIDFilter``(org.dvb.application.AppID appID)` Construit le filtre sur la base d'un identifiant `AppID` particulier.

Résumé pour la méthode	
`boolean`	**`accept`**`(RecordingRequest entry)` Vérifie si la `RecordingRequest` donnée passe le filtre.
`org.dvb.application.AppID`	**`getFilterValue`**`()` Rapporte la valeur de `AppID` utilisée pour créer ce filtre.

Méthodes héritées de la classe org.ocap.shared.dvr.navigation.RecordingListFilter
`setCascadingFilter`

Méthodes héritées de la classe java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Détail pour le constructeur

AppIDFilter

```
public AppIDFilter(org.dvb.application.AppID appID)
```

Construit le filtre sur la base d'un `AppID` particulier.

Paramètre:

`appID` – valeur d'AppID pour confronter les `RecordingRequest`.

Détail pour la méthode

getFilterValue

```
public org.dvb.application.AppID getFilterValue()
```

Rapporte la valeur de l'`AppID` utilisé pour créer ce filtre.

Retourne:

la valeur de l'`AppID` utilisé pour créer ce filtre.

---

accept

```
public boolean accept(RecordingRequest entry)
```

Vérifie si la `RecordingRequest` donnée passe le filtre.

Spécifié par:

`accept` dans la classe `RecordingListFilter`

Paramètres:

`entry` – Une `RecordingRequest` individuelle à évaluer par rapport à l'algorithme de filtrage.

Retourne:

`vrai` si la `RecordingRequest` est du type indiqué par la valeur du filtre; `faux` autrement.

D.2.3 Classe OrgIdFilter

```
org.ocap.shared.dvr.navigation
        Class OrgIDFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
       +--org.ocap.shared.dvr.navigation.AppIDFilter
```

---

classe publique **OrgIDFilter**

étend RecordingListFilter

Filtre à filtre sur la base de l'identifiant OrgID.

Résumé pour le constructeur
`OrgIDFilter``(int orgID)` Construit le filtre sur la base d'un identifiant d'organisation particulier.

Résumé pour la méthode	
`boolean`	**`accept`**`(RecordingRequest entry)` Vérifie si la `RecordingRequest` donnée passe le filtre.
`int`	**`getFilterValue`**`()` Rapporte la valeur de l'identifiant d'organisation utilisé pour créer ce filtre.

Méthodes héritées de la classe org.ocap.shared.dvr.navigation.RecordingListFilter
`setCascadingFilter`

Méthodes héritées de la classe java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Détail pour le constructeur

OrgIDFilter

`public` **`OrgIDFilter`**`(int orgID)`

Construit le filtre sur la base d'un identifiant d'organisation particulier.

Paramètres:

`orgID` – valeur de l'identifiant d'organisation pour confronter les instances de `RecordingRequest`.

Détail pour la méthode

getFilterValue

`public int` **`getFilterValue`**`()`

Rapporte la valeur de l'identifiant d'organisation utilisé pour créer ce filtre.

Retourne:

L'identifiant d'organisation utilisé pour filtrer.

---

accept

```
public boolean accept(RecordingRequest entry)
```

Vérifie si la `RecordingRequest` donnée passe le filtre.

Spécifié par:

`accept` dans la classe `RecordingListFilter`

Paramètres:

`entry` – Une `RecordingRequest` individuelle à évaluer par rapport à l'algorithme de filtrage.

Retourne:

`vrai` si la `RecordingRequest` est du type indiqué par la valeur du filtre; `faux` autrement.

D.2.4 Interface RecordingList

```
org.ocap.shared.dvr.navigation
        Interface RecordingList
```

---

interface publique **RecordingList**

RecordingList représente une liste d'enregistrements.

Résumé pour la méthode	
boolean	**contains**(RecordingRequest entry) Vérifie si l'objet `RecordingRequest` indiqué est contenu dans la liste.
RecordingListIterator	**createRecordingListIterator**() Génère un itérateur sur les éléments `RecordingRequest` dans la liste.
RecordingList	**filterRecordingList**(RecordingListFilter filter) Crée un nouvel objet `RecordingList` qui est un sous-ensemble de cette liste, sur la base des conditions spécifiées par un objet `RecordingListFilter`.
RecordingRequest	**getRecordingRequest**(int index) Rapporte la `RecordingRequest` à la position d'indice spécifiée.
int	**indexOf**(RecordingRequest entry) Rapporte la position de la première occurrence de l'objet `RecordingRequest` indiqué dans la liste.

int	**size**() Rapporte le nombre d'objets RecordingRequest dans la liste.
RecordingList	**sortRecordingList**(RecordingListComparator sortCriteria) Crée une nouvelle RecordingList qui contient tous les éléments de cette liste triés conformément aux critères spécifiés par un RecordingListComparator.

Détail pour la méthode

filterRecordingList

public RecordingList **filterRecordingList**(RecordingListFilter filter)

Crée un nouvel objet RecordingList qui est un sous-ensemble de cette liste, sur la base des conditions spécifiées par un objet RecordingListFilter. Cette méthode peut être utilisée pour générer des listes de spécialisation croissante d'objets RecordingRequest sur la base de critères de filtrage multiples. Si le filtre est null, la RecordingList résultante sera un duplicata de cette liste.

Noter que la méthode accept du RecordingListFilter donné sera invoquée pour chaque RecordingRequest à filtrer en utilisant le même fil d'application qui invoque cette méthode.

Paramètres:

filter – Un filtre qui impose des contraintes à la liste d'enregistrements demandée, ou null.

Retourne:

Un objet RecordingList créé sur la base des règles de filtrage spécifiées.

---

createRecordingListIterator

public RecordingListIterator **createRecordingListIterator**()

Génère un itérateur sur les éléments RecordingRequest dans cette liste.

Retourne:

Un RecordingListIterator sur les RecordingRequest dans cette liste.

---

contains

public boolean **contains**(RecordingRequest entry)

Vérifie si l'objet RecordingRequest indiqué est contenu dans la liste.

Paramètres:

entry – L'objet RecordingRequest à rechercher.

Retourne:

vrai si la RecordingRequest spécifiée est membre de la liste; faux autrement.

---

indexOf

`public int indexOf(RecordingRequest entry)`

Rapporte la position de la première occurrence de l'objet `RecordingRequest` indiqué dans la liste.

Paramètres:

`entry` – L'objet `RecordingRequest` à rechercher.

Retourne:

L'indice de la première occurrence de `entry`, ou `-1` si `entry` n'est pas contenu dans la liste.

---

size

`public int size()`

Rapporte le nombre d'objets `RecordingRequest` dans la liste.

Retourne:

Le nombre d'objets `RecordingRequest` dans la liste.

---

getRecordingRequest

`public RecordingRequest getRecordingRequest(int index)`

Rapporte la `RecordingRequest` à la position d'indice spécifiée.

Paramètres:

`index` – Une position dans la `RecordingList`.

Retourne:

La `RecordingRequest` à l'indice spécifié.

Invocation:

`java.lang.IndexOutOfBoundsException` – Si `index` < 0 ou si `index` > `size()-1`.

---

sortRecordingList

`public RecordingList sortRecordingList(RecordingListComparator sortCriteria)`

Crée une nouvelle `RecordingList` qui contient tous les éléments de cette liste triés conformément aux critères spécifiés par un `RecordingListComparator`.

Paramètres:

`sortCriteria` – le critère de tri à appliquer pour trier les entrées dans la liste d'enregistrements.

Retourne:

Une copie triée de la liste des enregistrements.

D.2.5 Interface RecordingListComparator

```
org.ocap.shared.dvr.navigation
        Interface RecordingListComparator
```

---

interface publique **RecordingListComparator**

Cette interface représente un critère de tri à appliquer lors du tri d'une RecordingList.

Résumé pour la méthode	
int	**compare**(RecordingRequest first, RecordingRequest second) Compare deux entrées pour vérifier si la première entrée devrait être placée avant la seconde entrée dans la liste de l'itérateur.

Détail pour la méthode

compare

```
public int compare(RecordingRequest first,
                   RecordingRequest second)
```

Compare deux entrées pour vérifier si la première entrée devrait être placée avant la seconde entrée dans la liste de l'itérateur.

Paramètres:

`first` – la première entrée à comparer

`second` – la seconde entrée à comparer

Retourne:

un entier positif si le premier argument devrait être placé avant le second argument; entier négatif si le second argument devrait être placé avant la première entrée; zéro si l'ordre actuel doit être conservé.

D.2.6 Classe RecordingListFilter

```
org.ocap.shared.dvr.navigation
        Class RecordingListFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
```

Sous-classes connues directement:

AppIDFilter, OrgIDFilter, RecordingStateFilter

---

classe abstraite publique **RecordingListFilter**

étend java.lang.Object

Classe de base pour tous les RecordingListFilter. Les sous-classes de RecordingListFilter peuvent être utilisées pour créer des filtres pour spécifier des restrictions.

Résumé pour le constructeur	
protected	**RecordingListFilter**() Construit le filtre.

Résumé pour la méthode	
abstract boolean	**accept**(RecordingRequest entry) Vérifie si une entrée particulière passe ce filtre.
void	**setCascadingFilter**(RecordingListFilter filter) Fournit un moyen de mettre les filtres en cascade.

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Détail pour le constructeur

RecordingListFilter

protected **RecordingListFilter**()

Construit le filtre.

Détail pour la méthode

accept

public abstract boolean **accept**(RecordingRequest entry)

Vérifie si une entrée particulière passe ce filtre. Les sous-types de RecordingListFilter outrepassent cette méthode pour fournir la logique d'une opération de filtrage sur les objets RecordingRequest individuels.

Paramètres:

entry – Une RecordingRequest à évaluer par rapport à l'algorithme de filtrage.

Retourne:

vrai si entry satisfait à l'algorithme de filtrage; faux autrement.

---

setCascadingFilter

public void **setCascadingFilter**(RecordingListFilter filter)

Donne le moyen de mettre les filtres en cascade. La méthode accept de ce filtre n'est invoquée que pour les entrées qui satisfont au filtre spécifié. Plusieurs invocations de cette méthode remplaceront le filtre précédemment établi.

Paramètres:

`filter` – le filtre qui sera appliqué avant de choisir les entrées pour lesquelles la méthode accept() est invoquée. Si le filtre actuel est dans la chaîne en cascade du filtre entré comme argument, cette méthode est sans effet.

D.2.7 Interface RecordingListIterator

```
org.ocap.shared.dvr.navigation
       Interface RecordingListIterator
```

---

interface publique **RecordingListIterator**

Cet itérateur pourrait être utilisé pour traverser les entrées dans une liste RecordingList.

Résumé pour la méthode	
`RecordingRequest`	**`getEntry`**`(int index)` Obtient l'objet `RecordingRequest` à la position spécifiée.
`int`	**`getPosition`**`()` Obtient la position actuelle du `RecordingListIterator`.
`int`	**`getPosition`**`(RecordingRequest entry)` Obtient la position d'une demande d'enregistrement spécifiée dans la liste.
`RecordingList`	**`getRecordingList`**`()` Obtient la liste d'enregistrements correspondant à ce RecordingListIterator.
`boolean`	**`hasNext`**`()` Vérifie si il y a une `RecordingRequest` à la position suivante de la liste.
`boolean`	**`hasPrevious`**`()` Vérifie si il y a une `RecordingRequest` à la position précédente de la liste.
`RecordingRequest[]`	**`nextEntries`**`(int n)` Obtient les 'n' prochains objets `RecordingRequest` de la liste.
`RecordingRequest`	**`nextEntry`**`()` Obtient le prochain objet `RecordingRequest` de la liste.
`RecordingRequest[]`	**`previousEntries`**`(int n)` Obtient les 'n' précédents objets `RecordingRequest` de la liste.
`RecordingRequest`	**`previousEntry`**`()` Obtient le précédent objet `RecordingRequest` de la liste.
`void`	**`setPosition`**`(int index)` Etablit la position actuelle du RecordingListIterator.
`void`	**`toBeginning`**`()` Remet l'itérateur au début de la liste, de sorte que `hasPrevious()` retourne `faux` et `nextEntry()` retourne la première `RecordingRequest` de la liste (si la liste n'est pas vide).

void	**toEnd**() Règle l'itérateur à la fin de la liste, de sorte que hasNext() retourne faux et que previousEntry() retourne la dernière RecordingRequest dans la liste (si la liste n'est pas vide).

Détail pour la méthode

toBeginning

public void **toBeginning**()

Remet l'itérateur au début de la liste, de sorte que hasPrevious() retourne faux et que nextEntry() retourne la première RecordingRequest de la liste (si la liste n'est pas vide).

---

toEnd

public void **toEnd**()

Met l'itérateur à la fin de la liste, de sorte que hasNext() retourne faux et previousEntry() retourne la dernière RecordingRequest dans la liste (si la liste n'est pas vide).

---

nextEntry

public RecordingRequest **nextEntry**()

Obtient le prochain objet RecordingRequest de la liste. Cette méthode peut être invoquée de façon répétée pour itérer à travers la liste.

Retourne:

L'objet RecordingRequest à la position suivante sur la liste.

Invocation:

java.util.NoSuchElementException – Si l'itération n'a pas de RecordingRequest suivante.

---

previousEntry

public RecordingRequest **previousEntry**()

Obtient l'objet RecordingRequest précédent sur la liste. Cette méthode peut être invoquée de façon répétée pour itérer à travers la liste en sens inverse.

Retourne:

L'objet RecordingRequest à la position précédente sur la liste.

Invocation:

java.util.NoSuchElementException – Si l'itération n'a pas de RecordingRequest précédente.

---

hasNext

`public boolean **hasNext**()`

Vérifie si il y a une `RecordingRequest` à la position suivante sur la liste.

Retourne:

`vrai` si il y a une `RecordingRequest` à la position suivante sur la liste; `faux` autrement.

---

hasPrevious

`public boolean **hasPrevious**()`

Vérifie si il y a une `RecordingRequest` à la position précédente sur la liste.

Retourne:

`vrai` si il y a une `RecordingRequest` à la position précédente sur la liste; `faux` autrement.

---

nextEntries

`public RecordingRequest[] **nextEntries**(int n)`

Obtient les 'n' objets `RecordingRequest` suivants sur la liste. Cette méthode avance aussi la position actuelle au sein de la liste. Si le nombre d'entrées demandé n'est pas disponible, les éléments restants sont retournés. Si la position actuelle est à la fin de l'itérateur, cette méthode retourne un dispositif de longueur zéro.

Paramètres:

`n` – le nombre de nouvelles entrées demandé.

Retourne:

un dispositif contenant les 'n' prochains objets `RecordingRequest` à partir de la position actuelle sur la liste.

---

previousEntries

`public RecordingRequest[] **previousEntries**(int n)`

Obtient les 'n' précédents objets `RecordingRequest` de la liste. Cette méthode change aussi la position actuelle au sein de la liste. Si le nombre d'entrées demandé n'est pas disponible, les éléments restants sont retournés. Si la position actuelle est au début de l'itérateur, cette méthode retourne un dispositif de longueur zéro.

Paramètres:

`n` – le nombre d'entrées précédentes demandé.

Retourne:

un dispositif contenant les 'n' précédents objets `RecordingRequest` à partir de la position actuelle dans la liste.

---

getEntry

```
public RecordingRequest getEntry(int index)
```

Obtient l'objet `RecordingRequest` à la position spécifiée. Cette méthode n'avance pas la position actuelle au sein de la liste.

Paramètres:

`index` – la position de la RecordingRequest à restituer.

Retourne:

la `RecordingRequest` à la position spécifiée.

Invocation:

`java.lang.IndexOutOfBoundsException` – si l'indice est supérieur à la taille de la liste.

---

getPosition

```
public int getPosition(RecordingRequest entry)
```

Obtient la position d'une demande d'enregistrement spécifiée dans la liste.

Paramètres:

`entry` – La demande d'enregistrement pour laquelle on recherche la position.

Retourne:

La position de l'enregistrement spécifié; –1 si l'entrée n'est pas trouvée.

---

getPosition

```
public int getPosition()
```

Obtient la position actuelle du RecordingListIterator. Ce serait la position à partir de laquelle la prochaine RecordingRequest serait restituée lorsqu'une application invoque la nextEntry.

Retourne:

la position actuelle du RecordingListIterator.

---

setPosition

```
public void setPosition(int index)
                 throws java.lang.IndexOutOfBoundsException
```

Règle la position actuelle du RecordingListIterator. Ce serait la position à partir de laquelle la prochaine RecordingRequest serait restituée lorsqu'une application invoque la prochaine nextEntry.

Paramètres:

`index` – la position actuelle du RecordingListIterator serait réglée à cette valeur.

Invocation:

`java.lang.IndexOutOfBoundsException` – si l'indice est supérieur à la taille de la liste.

---

getRecordingList

```
public RecordingList getRecordingList()
```

Obtient la liste d'enregistrement correspondant à ce RecordingListIterator.

Retourne:

la RecordingList correspondant à cet itérateur.

D.3 Paquetage de media partagé

```
org.ocap.shared.media
        Package
```


D.3.1 Interface TimeShiftControl

```
org.ocap.shared.media
        Interface TimeShiftControl
```

Toutes les super interfaces:

javax.media.Control

---

interface publique **TimeShiftControl**

étend javax.media.Control

Cette interface représente un contrôle de mode évolué qui peut être utilisé pour restituer plus d'informations correspondant à la lecture de la mémoire tampon de différé. Ce contrôle ne sera disponible que si le service présenté sur le contexte de service est un service de diffusion et si il y a une mémoire tampon de différé associée au contexte de service.

Résumé pour la méthode	
`javax.media.Time`	**`getBeginningOfBuffer`**`()` Obtient l'heure de support correspondant au début en cours de la mémoire tampon de différé.
`javax.media.Time`	**`getDuration`**`()` Obtient la durée du contenu qui est actuellement dans la mémoire tampon de différé.

javax.media.Time	**getEndOfBuffer**() Obtient l'heure de support correspondant à la fin de la mémoire tampon de différé.
javax.media.Time	**getMaxDuration**() Obtient la valeur estimée de la durée maximum de contenu qui pourrait être mise en mémoire tampon en utilisant cette mémoire tampon de différé.

Méthodes héritées de l'interface javax.media.Control
getControlComponent

Détail pour la méthode

getBeginningOfBuffer

public javax.media.Time **getBeginningOfBuffer**()

Obtient l'heure de support correspondant au début actuel de la mémoire tampon de différé. Cela pourrait être l'heure de support correspondant au début de la mémoire tampon, avant que la mémoire tampon ne recommence au début, ou l'heure de support correspondant au début de la zone de mémoire tampon valide après le redémarrage au début.

Retourne:

l'heure de support correspondant au début de la mémoire tampon de différé.

---

getEndOfBuffer

public javax.media.Time **getEndOfBuffer**()

Obtient l'heure de support correspondant à la fin de la mémoire tampon de différé. Cela peut être l'heure du système en cours si l'enregistrement en différé est toujours en cours ou l'heure de support correspondant au point de fin pour la zone valide de la mémoire tampon de différé.

Retourne:

l'heure de support correspondant à la fin de la mémoire tampon de différé.

---

getDuration

public javax.media.Time **getDuration**()

Obtient la durée du contenu qui est actuellement dans la mémoire tampon de différé. La valeur retournée est la durée du contenu lorsqu'il est lu à un taux de 1,0.

Retourne:

Un objet Time qui représente la durée.

---

getMaxDuration

```
public javax.media.Time getMaxDuration()
```

Obtient la valeur estimée de la durée maximum du contenu qui pourrait être mis dans la mémoire tampon de différé en utilisant cette mémoire tampon de différé. La valeur retournée est la durée du contenu lu à un taux de 1,0.

Retourne:

Un objet Time qui représente la valeur maximum de la durée.

D.3.2 Classe BeginningOfContentEvent

```
org.ocap.shared.media
        Class BeginningOfContentEvent

java.lang.Object
  +--java.util.EventObject
      +--javax.media.ControllerEvent
          +--javax.media.RateChangeEvent
              +--org.ocap.shared.media.BeginningOfContentEvent
```

Toutes les interfaces implémentées:

javax.media.MediaEvent, java.io.Serializable

---

classe publique **BeginningOfContentEvent**

étend javax.media.RateChangeEvent

BeginningOfContentEvent est un événement RateChangeEvent qui est affiché lorsque le changement de taux est dû à un rembobinage qui touche le début du support, ou à la mémoire tampon de différé qui atteint la profondeur maximum.

Voir aussi:

Forme en série

Résumé du champ

Champs hérités de la classe java.util.EventObject
`Source`

Résumé pour le constructeur
`BeginningOfContentEvent``(javax.media.Controller from, float newRate)` Crée un événement BeginningOfContentEvent.

Méthodes héritées de la classe javax.media.RateChangeEvent
`getRate, toString`

Méthodes héritées de la classe javax.media.ControllerEvent
getSource, getSourceController

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail pour le constructeur

BeginningOfContentEvent

```
public BeginningOfContentEvent(javax.media.Controller from,
                               float newRate)
```

Crée un événement BeginningOfContentEvent.

Paramètres:

`from` – le contrôleur qui génère l'événement.

D.3.3 Classe EndOfContentEvent

```
org.ocap.shared.media
       Class EndOfContentEvent

java.lang.Object
  +--java.util.EventObject
      +--javax.media.ControllerEvent
          +--javax.media.RateChangeEvent
              +--org.ocap.shared.media.EndOfContentEvent
```

Toutes les interfaces implémentées:

javax.media.MediaEvent, java.io.Serializable

---

classe publique **EndOfContentEvent**

étend javax.media.RateChangeEvent

EndOfContentEvent est un événement RateChangeEvent qui est affiché lorsque le changement de taux est dû à la lecture avant qui heurte la fin du contexte mémorisé, ou à une lecture avant qui rattrape le point d'enregistrement en direct.

Voir aussi:

Forme en série

Résumé du champ

Champs hérités de la classe java.util.EventObject
source

Résumé pour le constructeur
EndOfContentEvent(javax.media.Controller from, float newRate) Crée un événement EndOfContentEvent.

Méthodes héritées de la classe javax.media.RateChangeEvent
getRate, toString

Méthodes héritées de la classe javax.media.ControllerEvent
getSource, getSourceController

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail pour le constructeur

EndOfContentEvent

```
public EndOfContentEvent(javax.media.Controller from,
                         float newRate)
```

Crée un événement EndOfContentEvent.

Paramètres:

`from` – le contrôleur qui génère l'événement.

D.3.4 Classe EnteringLiveModeEvent

```
org.ocap.shared.media
       Class EnteringLiveModeEvent

java.lang.Object
  +--java.util.EventObject
      +--javax.media.ControllerEvent
          +--org.ocap.shared.media.EnteringLiveModeEvent
```

Toutes les interfaces implémentées:

javax.media.MediaEvent, java.io.Serializable

---

classe publique **EnteringLiveModeEvent**

étend javax.media.ControllerEvent

EnteringLiveModeEvent est un événement ControllerEvent qui est affiché lorsque le contrôleur a commencé la lecture d'un flux de diffusion en direct. Cet événement est envoyé à un ControllerListener enregistré en plus de tout RateChangeEvent ou MediaTimeSetEvent.

Voir aussi:

Forme en série

Résumé du champ

Champs hérités de la classe java.util.EventObject

`source`

Résumé pour le constructeur

`EnteringLiveModeEvent``(javax.media.Controller from)`
Crée un événement EnteringLiveModeEvent.

Méthodes héritées de la classe javax.media.ControllerEvent

`getSource, getSourceController, toString`

Méthodes héritées de la classe java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Détail pour le constructeur

EnteringLiveModeEvent

`public` **`EnteringLiveModeEvent`**`(javax.media.Controller from)`

Crée un EnteringLiveModeEvent.

Paramètres:

`from` – le contrôleur qui génère l'événement.

D.3.5 Classe LeavingLiveModeEvent

```
org.ocap.shared.media
        Class LeavingLiveModeEvent

java.lang.Object
  +--java.util.EventObject
      +--javax.media.ControllerEvent
          +--org.ocap.shared.media.LeavingLiveModeEvent
```

Toutes les interfaces implémentées:

javax.media.MediaEvent, java.io.Serializable

---

classe publique **LeavingLiveModeEvent**

étend javax.media.ControllerEvent

LeavingLiveModeEvent est un événement ControllerEvent qui est affiché lorsque le contrôleur ne lit plus un flux de diffusion en direct. Cet événement est envoyé à un ControllerListener enregistré en plus de tout RateChangeEvent ou MediaTimeSetEvent.

Voir aussi:

Forme en série

Résumé du champ

Champs hérités de la classe java.util.EventObject

`source`

Résumé pour le constructeur

`LeavingLiveModeEvent``(javax.media.Controller from)`
Crée un événement LeavingLiveModeEvent.

Méthodes héritées de la classe javax.media.ControllerEvent

`getSource, getSourceController, toString`

Méthodes héritées de la classe java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Détail pour le constructeur

LeavingLiveModeEvent

`public` **`LeavingLiveModeEvent`**`(javax.media.Controller from)`

Crée un événement LeavingLiveModeEvent.

Paramètres:

`from` – le contrôleur qui génère l'événement.

D.3.6 Interface MediaTimeFactoryControl

```
org.ocap.shared.media
        Interface MediaTimeFactoryControl
```

Toutes les super interfaces:

javax.media.Control

interface publique **MediaTimeFactoryControl**

étend javax.media.Control

Donne la capacité d'obtenir des heures de support avec des caractéristiques particulières variées lorsqu'elles sont appliquées au contenu en cours de lecture par ce lecteur JMF. Le comportement de ces heures de support dépend de l'implémentation si elles sont utilisées avec un autre lecteur JMF.

Résumé pour la méthode	
`javax.media.Time`	**`getRelativeTime`**`(long offset)` Obtient une heure de support par rapport à la localisation en cours.
`javax.media.Time`	**`setTimeApproximations`**`(javax.media.Time original, boolean beforeOrAfter)` Permet aux applications de contrôler précisément la position où commence la lecture à la suite de l'invocation de Player.setMediaTime.

Méthodes héritées de l'interface javax.media.Control
`getControlComponent`

Détail pour la méthode

getRelativeTime

```
public javax.media.Time getRelativeTime(long offset)
```

Obtient une heure de support par rapport à la localisation en cours.

Paramètres:

`offset` – le décalage par rapport à la localisation en cours mesurée en nanosecondes.

Retourne:

une heure de support

---

setTimeApproximations

```
public javax.media.Time setTimeApproximations(javax.media.Time original,
                                              boolean beforeOrAfter)
```

Permet aux applications de contrôler précisément la position où commence la lecture à la suite de l'invocation de Player.setMediaTime. Cette méthode prend en entrée une heure de support originale et retourne une nouvelle heure de support qui encapsule l'heure de support originale et une indication de la façon dont l'heure de support originale est à interpréter lorsqu'elle est utilisée dans une invocation de Player.setMediaTime.

Paramètres:

`original` – l'heure de support originale

`beforeOrAfter` – si il est vrai, l'heure de support à laquelle commence la lecture sera à l'heure originale ou avant (c'est-à-dire que la présentation en lecture du contenu est garantie à l'heure de support originale). Si il est faux, l'heure de support à laquelle commence la lecture sera après l'heure originale (c'est-à-dire que ni le contenu à l'heure de support originale ni aucun contenu antérieur à cette heure originale ne sera présenté en lecture).

Retourne:

une nouvelle heure de support.

D.3.7 Interface TimeLine

```
org.ocap.shared.media
        Interface TimeLine
```

---

interface publique **TimeLine**

Représente une ligne horaire transmise. Les lignes horaires transmises commencent à une heure de support au sein d'une partie de contenu et finissent à une heure de support ultérieure dans ce contenu. Les lignes horaires transmises sont valides à toutes les heures de support entre ces points. Elles augmentent de façon linéaire ou sont en pause. La valeur d'une ligne horaire transmise ne connaît aucune discontinuité.

Résumé pour la méthode	
`javax.media.Time`	**`getFirstMediaTime`**`()` Retourne la première heure de support à laquelle cette ligne horaire est valide.
`long`	**`getFirstTime`**`()` Retourne la première heure valide dans cette ligne horaire.
`javax.media.Time`	**`getLastMediaTime`**`()` Retourne la dernière heure à laquelle cette ligne horaire est valide.
`long`	**`getLastTime`**`()` Retourne la dernière heure valide dans cette ligne horaire.
`javax.media.Time`	**`getMediaTime`**`(long time)` Traduit une heure de cette ligne horaire en l'heure support correspondante.
`long`	**`getTime`**`(javax.media.Time mediatime)` Traduit une heure support en l'heure correspondante dans cette ligne horaire.

Détail pour la méthode

getFirstMediaTime

```
public javax.media.Time getFirstMediaTime()
                                  throws TimeLineInvalidException
```

Retourne la première heure de support à laquelle cette ligne horaire est valide. Pour un enregistrement programmé, c'est le premier point au sein de la partie de contenu où la ligne horaire est valide. Pour un enregistrement en différé, si la ligne horaire commence dans la mémoire tampon de différé, l'heure de support à laquelle elle commence sera retournée. Si la

ligne horaire commence avant le début de la mémoire tampon de différé, l'heure de support du début de la mémoire tampon de différé sera retournée. Noter que si la mémoire tampon de différé est pleine et que l'enregistrement en différé est en cours, le début de la mémoire tampon va se déplacer au fur et à mesure de la réécriture des nouvelles données par dessus l'ancien début de la mémoire tampon.

Retourne:

une heure de support.

invocation:

`TimeLineInvalidException` – si la ligne horaire n'est plus valide dans cette partie de contenu. Par exemple, la partie de contenu est un enregistrement en différé et la fin de la ligne horaire n'est plus dans les limites de la mémoire tampon.

---

getLastMediaTime

```
public javax.media.Time getLastMediaTime()
                                  throws TimeLineInvalidException
```

Retourne la dernière heure à laquelle cette ligne horaire est valide. Pour un enregistrement programmé, c'est le dernier point au sein de la partie de contenu où la ligne horaire est valide. Pour un enregistrement en différé, si la ligne horaire se termine au sein de la mémoire tampon de différé, l'heure de support à laquelle elle commence sera retournée. Si la ligne horaire se termine après la fin de la mémoire tampon de différé, l'heure de support de la fin de la mémoire tampon de différé sera retournée. Noter que si la mémoire tampon de différé est pleine et si un enregistrement en différé est en cours, la fin de la mémoire tampon se déplacera au fur et à mesure que les données nouvellement écrites recouvrent l'ancien début de la mémoire tampon.

Retourne:

une heure de support.

Invocation:

`TimeLineInvalidException` – si la ligne horaire n'est plus valide dans cette partie de contenu. Par exemple, la partie de contenu est un enregistrement en différé et la fin de la ligne horaire n'est plus dans les limites de la mémoire tampon.

---

getFirstTime

```
public long getFirstTime()
                  throws TimeLineInvalidException
```

Retourne la première heure valide dans cette ligne horaire. Pour un enregistrement programmé, c'est le premier point au sein ce cette partie de contenu où la ligne horaire est valide. Pour un enregistrement en différé, si la ligne horaire commence dans les limites de la mémoire tampon de différé, l'heure à laquelle il commence sera retournée. Si la ligne horaire commence avant le début de la mémoire tampon de différé, l'heure du début de la mémoire tampon de différé sera retournée. Noter que si la mémoire tampon de différé est pleine et que l'enregistrement en différé est en cours, le début de la mémoire tampon se déplacera au fur et à mesure que les nouvelles données écrites recouvrent l'ancien début de la mémoire tampon.

Retourne:

une heure dans cette ligne horaire.

Invocation:

`TimeLineInvalidException` – si la ligne horaire n'est plus valide dans cette partie de contenu. Par exemple, la partie de contenu est un enregistrement en différé et la fin de la ligne horaire n'est plus dans les limites de la mémoire tampon.

---

getLastTime

```
public long getLastTime()
                throws TimeLineInvalidException
```

Retourne la dernière heure valide dans cette ligne horaire. Pour un enregistrement programmé, c'est le dernier point au sein de la partie de contenu où la ligne horaire est valide. Pour un enregistrement en différé, si la ligne horaire se termine au sein de la mémoire tampon de différé, l'heure de support à laquelle il se termine sera retournée. Si l'heure de support se termine après la fin de la mémoire tampon de différé, l'heure de support de la fin de la mémoire tampon de différé sera retournée. Noter que si la mémoire tampon de différé est pleine et que l'enregistrement en différé est en cours, la fin de la mémoire tampon se déplacera au fur et à mesure que des données nouvellement écrites se substituent à l'ancien début de la mémoire tampon.

Retourne:

une heure dans cette ligne horaire.

Invocation:

`TimeLineInvalidException` – si la ligne de temps n'est plus valide dans cette partie de contenu. Par exemple, la partie de contenu est un enregistrement en différé et la fin de la ligne horaire n'est plus dans les limites de la mémoire tampon.

---

getMediaTime

```
public javax.media.Time getMediaTime(long time)
                              throws TimeLineInvalidException,
                                     TimeOutOfRangeException
```

Traduit une heure de cette ligne horaire dans l'heure support correspondante. Si l'heure est une de celles où la ligne horaire marque une pause, l'heure support retournée devra être la plus forte heure de support correspondante à l'heure spécifiée.

Paramètres:

`time` – une heure dans cette ligne horaire

Retourne:

l'heure de support correspondante.

Invocation:

`TimeLineInvalidException` – si la ligne horaire n'est plus valide dans cette partie de contenu. Par exemple, la partie de contenu est un enregistrement en différé et la fin de la ligne horaire n'est plus dans les limites de la mémoire tampon.

`TimeOutOfRangeException` – si l'heure spécifiée n'est pas dans cette ligne horaire.

getTime

```
public long getTime(javax.media.Time mediatime)
             throws TimeLineInvalidException,
                    TimeOutOfRangeException
```

Traduit une heure de support en heure correspondante dans cette ligne horaire.

Paramètres:

`mediatime` – une heure de support.

Retourne:

L'heure correspondante dans cette ligne horaire.

Invocation:

`TimeLineInvalidException` – si la ligne horaire n'est plus valide dans cette partie de contenu. Par exemple, la partie de contenu est un enregistrement programmé et la fin de la ligne horaire n'est plus dans les limites de la mémoire tampon.

`TimeOutOfRangeException` – si l'heure de support n'est pas dans cette ligne horaire.

D.3.8 Interface TimeLineControl

```
org.ocap.shared.media
        Interface TimeLineControl
```

Toutes les super interfaces:

javax.media.Control

---

interface publique **TimeLineControl**

étend javax.media.Control

Fournit l'accès aux lignes horaires transmises dans une partie de contenu.

Résumé pour la méthode	
`TimeLine[]`	**`getTimeLines`**`()` Retourne toutes les lignes horaires transmises trouvées dans une partie de contenu.

Méthodes héritées de l'interface javax.media.Control
`getControlComponent`

Détail pour la méthode

getTimeLines

```
public TimeLine[] getTimeLines()
```

Retourne toutes les lignes horaires transmises trouvées dans une partie de contenu. Si aucune ligne horaire transmise n'est présente, un dispositif de longueur 0 est retourné.

Retourne:

Un dispositif de lignes horaires.

D.3.9 Classe TimeLineInvalidException

```
org.ocap.shared.media
        Class TimeLineInvalidException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.media.TimeLineInvalidException
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **TimeLineInvalidException**

étend java.lang.Exception

Cette exception est retournée lorsqu'une ligne horaire n'est plus valide dans une partie de contenu pour laquelle elle a été obtenue. Par exemple, la partie de contenu est un enregistrement en différé et la fin de la ligne horaire n'est plus dans la mémoire tampon.

Voir aussi:

Forme en série.

Résumé pour le constructeur
TimeLineInvalidException() Construit une exception TimeLineInvalidException sans message de détail.

Méthodes héritées de la classe java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail pour le constructeur

TimeLineInvalidException

public **TimeLineInvalidException**()

Construit une exception TimeLineInvalidException sans message de détail.

D.3.10 Classe TimeOutOfRangeException

```
org.ocap.shared.media
        Class TimeOutOfRangeException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.media.TimeOutOfRangeException
```

Toutes les interfaces implémentées:

java.io.Serializable

---

classe publique **TimeOutOfRangeException**

étend java.lang.Exception

Cette exception est retournée lorsqu'une heure ou une heure de support est en dehors de la gamme valide pour une ligne horaire particulière.

Voir aussi:

Forme en série

Résumé pour le constructeur
TimeOutOfRangeException() Construit une exception TimeOutOfRangeException sans message de détail

Méthodes héritées de la classe java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Méthodes héritées de la classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Détail pour le constructeur

TimeOutOfRangeException

public **TimeOutOfRangeException**()

Construit une exception TimeOutOfRangeException sans message de détail

Bibliographie

[b-DAVIC 1.4.1p9] DAVIC 1.4.1p9, "DAVIC 1.4.1 Specification Part 9 – Information Representation" *(DAVIC 1.4.1 Spécification Partie 9 – Représentation des informations)*.

[b-ETSI TS 102 816] ETSI TS 102 816, "Digital Video Broadcasting (DVB); PVR/PDR Extension to the Multimedia Home Platform" *(Vidéodiffusion numérique (DVB); Extension PVR/PDR pour la plate-forme multimédia résidentielle)*.

NOTE – Au moment de la publication de la présente Recommandation, la référence ci-dessus n'est disponible que comme DVB Bluebook A088.rev1. Elle deviendra disponible en son temps auprès de l'ETSI.

[b-OC-SP-OCAP-DVR-I02-050524] OC-SP-OCAP-DVR-I02-050524, OpenCable Application Platform Specification; OCAP Digital Video Recorder (DVR) *(Spécification de la plate-forme d'application OpenCable; Magnétoscope numérique OCAP)*.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	**Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias**
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	Gestion des télécommunications y compris le RGT et maintenance des réseaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données, communication entre systèmes ouverts et sécurité
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2008