UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

T.172

(02/98)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE T: TERMINAUX DES SERVICES TÉLÉMATIQUES

MHEG-5 – Support des applications interactives de niveau de base

Recommandation UIT-T T.172

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE T

TERMINAUX DES SERVICES TÉLÉMATIQUES

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

RECOMMANDATION UIT-T T.172

MHEG-5 – SUPPORT DES APPLICATIONS INTERACTIVES DE NIVEAU DE BASE

Résumé

La présente Recommandation spécifie la sémantique et la syntaxe d'échange sous forme définitive des objets MHEG-5 en se basant sur les concepts définis dans la Recommandation T.171. Ces objets ont pour objectif d'être utilisés dans le domaine des applications client/serveur multimédias interactives, à savoir applications de Vidéo à la demande (ou presque) et applications de navigation et de visionnage.

Source

La Recommandation UIT-T T.172, élaborée par la Commission d'études 16 (1997-2000) de l'UIT-T, a été approuvée le 6 février 1998 selon la procédure définie dans la Résolution n° 1 de la CMNT.

La présente Recommandation est alignée techniquement sur l'ISO/CEI 13522-5.

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.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études 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 n° 1 de la CMNT.

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.

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 responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Page

Recommandation T.172

MHEG-5 – SUPPORT DES APPLICATIONS INTERACTIVES DE NIVEAU DE BASE

(Genève, 1998)

1 Domaine

La présente Recommandation spécifie la sémantique et la syntaxe d'échange sous forme définitive des objets MHEG-5, basées sur les concepts définis dans la Recommandation T.171. Ces objets ont pour objectif d'être utilisés dans des applications client/serveur multimédias interactives simples, à savoir des applications de Vidéo à la demande (ou presque) et des applications de navigation et de visionnage.

1.1 Spécificité du domaine

Etant donné que l'on s'attend à utiliser la présente Recommandation pour l'interopérabilité des applications inter-plates-formes, son domaine est limité à une définition précise et spécifique des classes MHEG-5. On reconnaît dans la présente Recommandation la sémantique induite par la spécification des objets MHEG-5 et par l'interprétation des comportements MHEG-5 à l'intérieur du système qui en fait usage.

1.2 Points ne rentrant pas dans le champ de la présente Recommandation

Le domaine exclut toute normalisation de modèles, services, systèmes, protocoles ou applications qui sont supposés faire usage des objets MHEG-5.

La représentation codée des données *content* ne rentre pas dans le domaine de la présente Recommandation.

2 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.

– Recommandation UIT-T X.680 (1994) | ISO/CEI 8824-1:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro 1: spécification de la notation de base.*

– Recommandation UIT-T X.690 (1994) | ISO/CEI 8825-1:1995, *Technologies de l'information – Règles de codage de la notation de syntaxe abstraite numéro un: spécification des règles de codage de base, des règles de codage canoniques et des règles de codage distinctives*.

– ISO/CEI 646:1991, *Technologies de l'information – Jeu ISO de caractères codés à 7 éléments pour l'échange d'informations.*

– RFC 1521 (1993), *MIME (Multipurpose Internet Mail Extensions) Partie 1*: *Mécanismes pour spécifier et décrire le format des corps de message Internet* (publié actuellement en anglais seulement).

3 Termes et définitions

La présente Recommandation définit les termes suivants:

3.1 classe abstraite: classe n'étant jamais instanciée dans un objet MHEG-5 échangeable.

NOTE – Une classe abstraite définit les attributs, comportements et sémantique des actions pouvant être échangées comme parties de tout objet MHEG-5 des sous-classes concrètes de cette classe abstraite.

3.2 action: ensemble d'actions élémentaires.

3.3 actif: état de tout objet MHEG-5 lorsque le comportement *Activation* s'est terminé avec succès pour cet objet.

Un objet actif possède son *RunningStatus* mis à *True*.

3.4 domaine applicatif: domaine spécifique des applications utilisant la présente Recommandation et fournissant des outils et valeurs supplémentaires afin de créer une instance concrète d'environnement MHEG-5.

NOTE – L'Appendice II fournit des informations supplémentaires sur les domaines applicatifs.

3.5 portée d'application: domaine commun à tous les objets MHEG-5 (*scenes* et *ingredients*) accédés à partir d'une application MHEG-5.

3.6 attribut: valeur typée et nommée attachée à une classe.

3.7 available (*disponible)*: état de tout objet MHEG lorsque le comportement *Preparation* s'est terminé avec succès pour cet objet.

Un objet *available* a son *AvailabilityStatus* mis à *True*.

3.8 classe de base: classe MHEG-5 définissant les attributs, comportements et sémantique des actions pouvant être partagées par une classe MHEG-5 donnée.

3.9 classe concrète: classe de tout objet MHEG-5 pouvant être codée et échangée conformément aux spécifications fournies dans les Annexes A ou B.

3.10 action élémentaire: représentation abstraite d'un message pouvant être envoyé à un objet.

NOTE – La présente Recommandation définit la sémantique des actions élémentaires disponibles pour chaque classe MHEG-5. Il faut noter que la classe Action (avec A majuscule) de MHEG-5 a une signification différente décrite dans le paragraphe 49.

3.11 événement: représentation abstraite d'une occurrence à signification spéciale pour tout objet MHEG-5.

NOTE – Les événements sont utilisés pour déclencher les conditions *Link* et pour faire valoir à l'extérieur l'exécution des séquences d'actions élémentaires.

3.12 attribut échangé: attribut faisant partie de la représentation codée échangeable d'un objet MHEG-5 et transmis avec cet objet.

3.13 inactif: état de tout objet MHEG-5 faisant suite à une exécution réussie du comportement *Deactivation* ou lorsque aucun comportement *Activation* n'a été appliqué avec succès sur cet objet.

Un objet MHEG-5 inactif possède son *RunningStatus* mis à *False*.

3.14 attribut hérité: attribut défini dans une classe de base de la classe de l'objet MHEG-5.

3.15 représentation échangeable: chaîne d'octets contenant les attributs échangés codés de cet objet MHEG-5 conformément à la syntaxe ASN.1 et codé conformément à l'Annexe A, ou bien lorsqu'une représentation textuelle est préférée, avec l'Annexe B.

3.16 attribut interne: structure de donnée abstraite, jamais échangée ou codée octet, utilisée pour définir la sémantique des comportements ou des actions internes de tout objet MHEG-5.

NOTE – Tout moteur MHEG-5 devrait considérer qu'un attribut interne fait partie de la représentation interne de l'objet MHEG-5. Cependant, ceci n'est pas obligatoire. Ce qui l'est est d'implémenter les fonctionnalités décrites par ces attributs internes.

3.17 comportement interne: fonction abstraite définissant la sémantique des actions élémentaires de toute classe MHEG-5.

NOTE – Un comportement interne de classe est la plupart du temps remplacé par les comportements internes des sous-classes de cette classe. Un moteur MHEG-5 pourrait considérer un comportement interne d'une classe comme une méthode privée de cette classe; cependant ceci n'est pas obligatoire.

3.18 application MHEG-5: ensemble de scènes et d'information de contrôle permettant à l'utilisateur de naviguer entre les scènes.

NOTE – La classe Application (avec A majuscule) a une signification différente et plus spécifique qui est donnée dans le paragraphe 10.

3.19 classe MHEG-5: définition abstraite des attributs internes échangés des parties d'objets multimédias et hypermédias échangeables ainsi qu'une définition de la sémantique des comportements internes et l'effet des actions MHEG-5 sur ces objets.

3.20 moteur MHEG-5: processus ou ensemble de processus interprétant les objets MHEG-5 codés conformément aux spécifications de codage définies dans les Annexes A ou B.

3.21 objet MHEG-5: instance de toute classe MHEG-5.

NOTE – Un objet MHEG-5 n'est pas un objet physique, mais plutôt une abstraction pouvant avoir plusieurs représentations de types différents. De nombreux services logiciels manipulent de telles représentations.

3.22 scène MHEG-5: structure assurant la coordination des présentations visuelles et audibles des objets MHEG-5.

3.23 classe mixte: classe abstraite n'héritant pas de la classe *Root*.

exemples: classe *Interactible*, classe *TokenManager*.

3.24 non-available *(non disponible)*: état de tout objet MHEG-5 lorsque le comportement *Destruction* s'est terminé avec succès ou lorsque aucun comportement *Preparation* n'a été appliqué avec succès sur cet objet.

Un objet *non-available* possède son *AvailabilityStatus* mis à *FALSE*. Même si aucun objet MHEG-5 n'existe dans le moteur MHEG-5, son *AvailabilityStatus* existe et vaut *FALSE*.

3.25 sous-classe: toute classe MHEG-5 partageant les mêmes attributs, comportements et sémantique d'actions d'une autre classe MHEG-5.

4 Conformité

Le présent paragraphe spécifie les prescriptions de conformité à appliquer aux moteurs MHEG-5 et aux applications MHEG-5.

4.1 Conformité des objets MHEG-5

Tout objet MHEG-5 aura une représentation octet. Pour les besoins d'échange, la représentation octet sera conforme à la syntaxe ASN.1 et au codage défini dans l'Annexe A, ou avec la grammaire de notation textuelle définie dans l'Annexe B. Le domaine applicatif choisira la représentation à

utiliser: celle de l'Annexe A ou celle de l'Annexe B, et cette représentation sera ensuite utilisée de manière exclusive dans le domaine applicatif.

Les attributs de tout objet MHEG-5 seront conformes à toutes les dispositions définies dans les sous-paragraphes pertinents de la présente Recommandation.

4.2 Conformité des moteurs MHEG-5

La conformité des moteurs MHEG-5 peut être seulement mesurée par rapport à une définition complète du domaine applicatif. Afin de spécifier complètement cette conformité, un domaine applicatif définira ce qui suit, en plus de la représentation d'échange:

1) un ensemble de classes à partir de la liste de toutes les classes de la présente Recommandation, comme prescrit au 4.2.1;
2) un ensemble d'attributs à partir de la liste donnée au 4.2.2;
3) des choix concrets supplémentaires comme ceux listés au 4.2.3.

NOTE – Se référer à l'Appendice II pour un exemple de définition complète de domaine applicatif.

4.2.1 Conformité pour l'acceptation d'un ensemble de Classes et d'Actions élémentaires

La conformité pour l'acceptation d'un ensemble de Classes et d'Actions élémentaires est définie comme suit.

Il est demandé à tout moteur MHEG-5 d'implémenter au moins l'ensemble minimum de classes suivant:

- classe Application

  tous les attributs, événements et comportements internes seront implémentés;
- classe Scène

  tous les attributs, événements et comportements internes seront implémentés;
- classe Lien

  tous les attributs, événements et comportements internes seront implémentés;
- classe Action

  tous les attributs, événements et comportements internes seront implémentés.

Tous les domaines applicatifs définiront une conformité à un ensemble de classes contenant au moins l'ensemble minimum décrit ci-dessus. Un domaine applicatif peut spécifier pour sa conformité un ensemble plus grand de classes et d'actions élémentaires; dans tous les cas le domaine applicatif listera clairement les classes et les actions élémentaires supportées.

Lorsque des classes supplémentaires sont supportées dans tout moteur MHEG-5, le moteur implémentera tous leurs attributs, événements, comportements internes et actions élémentaires comme définis dans la présente Recommandation, avec une exception possible concernant les attributs optionnels listés dans 4.2.2. Concernant la classe Action, le moteur implémentera tous les effets des actions MHEG-5 élémentaires correspondant à l'ensemble spécifié de classes. C'est le rôle de chaque domaine applicatif de choisir et de bien définir un ensemble de classes requis pour ce domaine applicatif spécifique.

Si une classe n'est pas supportée par un moteur MHEG-5 et si un objet de cette classe est envoyé au moteur MHEG-5, il y a production d'une erreur traitée par le traitement d'erreur par défaut défini au 53.7.

4.2.2 Conformité à un ensemble de fonctionnalités de moteur

La conformité à un ensemble de fonctionnalités de moteur est définie comme suit:

Tout moteur MHEG-5 fournira tous les mécanismes normatifs définis aux paragraphes 51 à 54.

Tout moteur MHEG-5 implémentera tous les effets des actions MHEG-5 et les comportements des classes MHEG-5 compris dans la définition de leur domaine applicatif, à l'exception des attributs optionnels suivants:

- connexions auxiliaires (correspondant aux actions *OpenConnection* et *CloseConnection*);
- cache (correspondant au cache des objets MHEG-5 et des données *content* des objets *Ingredients*);
- clonage (correspondant à l'action *Clone* définie dans la classe *Ingredient*);
- curseur à déplacement libre;
- mise à l'échelle de phototrame et de vidéo (correspondant aux actions *ScaleBitmap* et *ScaleVideo* des classes *Bitmap* et *Video*);
- empilement d'applications (correspondant à l'action *Spawn* de la classe *Application*);
- mode Trick (correspondant à l'action *SetSpeed* de la classe *Stream*).

Un domaine applicatif définira clairement la liste de ceux des attributs ci-dessus qui sont obligatoires ou optionnels pour ce qui concerne la conformité au domaine applicatif.

4.2.3 Besoins supplémentaires en matière de spécification de conformité

Pour définir complètement une conformité, les tableaux suivants seront spécifiés pour une application donnée, en plus des deux éléments ci-dessus.

NOTE – Pour chacun de ces tableaux, un exemple concret est donné dans l'Appendice II, définissant ainsi un domaine applicatif.

- codage de données *Content*

  Le domaine applicatif spécifiera quel type de données *Content* et quel type de codage sont supportés. Les deux tableaux suivants seront remplis pour chaque domaine applicatif considéré.

Types de données de contenu pris en charge

Attribut	Valeurs permises
FontAttribut	
FontName	
AbsoluteColour	
CharacterSet	
TransitionEffect	

Tableau de codage

Type de contenu	Codage de contenu	Valeur de crochet
format de codage de *Font*		
format de codage de *Palette*		
format de codage de *Bitmap*		
format de codage de *Text*		
format de codage de *EntryField*		
format de codage de *HyperText*		
format de codage de *Stream*		
format de codage de *LineArt*		
format de codage de *CursorShape*		
format de codage de *InterchangedProgram*		
format de codage de *AbsoluteColour*		

- Registres *UserInput*

  Pour les besoins opérationnels, le domaine applicatif spécifiera un ou plusieurs *InputEventRegisters*. Chaque registre a un numéro, échangé comme un des paramètres d'un objet *Scene* et un contenu (non échangé) comprenant un ensemble de numéros (représentant des *UserInputEventTag*) et un nom. Les doublets nom/numéro font correspondre un *UserInputEventTag* spécifique à un événement d'entrée logique. Il est laissé à la charge de l'implémenteur du moteur de faire correspondre l'événement d'entrée logique à un ou plusieurs événements d'entrée physiques.

  Le tableau suivant sera complété pour chaque domaine applicatif considéré.

Registre #	UserInputEventTag	Sémantique	Commentaire
(*Entier*)	…	…	…

- EngineEvent

  Un domaine applicatif MHEG-5 peut spécifier un ensemble de numéros associés avec *EngineEvent* pour s'y retrouver entre les événements externes variés conduisant à la génération de l'élément *EngineEvent*. Dans un tel cas, à chacun des *EngineEvent* spécifiques correspondra un entier dans un tableau comme ci-dessous. Les valeurs qui ne sont pas réservées par le domaine applicatif sont libres et peuvent être utilisées par le programmeur de l'application.

EngineEvent	EventTag
…	(*Entier*)

- *GetEngineSupport*

  Les domaines applicatifs peuvent définir, en plus des chaînes mentionnées dans la présente Recommandation, d'autres chaînes permises pour l'action *GetEngineSupport*. Dans un tel cas, le domaine applicatif listera clairement ces chaînes supplémentaires.

- Contraintes sémantiques sur les applications MHEG-5

  Pour chaque attribut défini par une chaîne dans l'action *GetEngineSupport*, un domaine applicatif peut choisir d'une certaine manière d'appliquer des contraintes sur ses applications. Si tel est le cas, un tableau de contraintes sera fourni.

Attribut	Contrainte
…	…

- Mappage de protocole et interaction externe

Finalement, certaines actions ont un effet externe à l'exécution. Ce sont des fonctions de récupération d'objets, de manipulation de flux et de fonctions d'appel externe. Pour assurer l'interopérabilité, ces actions auront un effet externe sous-jacent commun. Ceci implique habituellement l'existence d'un mappage cohérent de ces actions envers les fonctions externes de communication sous-jacentes pour les utilisateurs des domaines applicatifs MHEG-5. Les mappages suivants seront fournis par le domaine applicatif.

Entité MHEG-5	Mappage requis	Sémantique des structures MHEG-5 ayant besoin d'une spécification
OpenConnection, *CloseConnection*	mappage de protocoles de gestion de connexion (et éventuellement de gestion de session) dans le domaine applicatif	• dans *OpenConnection*: – *Protocol* – *Address*
objets *RemoteProgram*	mappage du protocole d'appel *RemoteProgram* dans le domaine applicatif	• dans *Call* et *Fork*: – *Name* – *Parameters* – *ProgramConnectionTag*
espace de nom d'Application	mappage de l'espace de nom dans le domaine applicatif	• *ObjectReference* • *ContentReference*
espace de nom d'Application dans le cas où l'action *TransitionTo* utilise le paramètre *ConnectionTag*	mappage de l'espace de nom dans le domaine applicatif	• *ObjectReference* • *ContentReference*
espace de nom de stockage permanent	mappage de l'espace de nom dans l'espace de stockage permanent	• dans *StorePersistent* et *ReadPersistent*: – *InFileName*, *OutFileName*
actions *Stream*	mappage de l'interface de flux dans le domaine applicatif	• dans *Stream* – *Speed* – *CounterPosition*
événements *Stream*	mappage des états de flux et aux événements flux dans le domaine applicatif	• dans *Stream* – *StreamPlaying*, *StreamStopped* (mappage de la machine d'état de flux du domaine applicatif) – *CounterPosition* – *StreamEventTag*

5 Aperçu général sur les classes MHEG-5

La présente Recommandation a été développée pour supporter la distribution des applications multimédias interactives selon une architecture client/serveur[1] à travers des plates-formes de différents types. La présente Recommandation définit une représentation sous forme définitive en vue d'un échange d'applications. Les applications sont composées principalement de code déclaratif, mais des dispositions pour l'appel de codes procéduraux ont été prises. Les applications MHEG-5 ont besoin d'être créées une seule fois et ensuite fonctionnent sur tout type de plate-forme conforme à la présente Recommandation. Les applications développées devraient être hébergées sur le serveur, et lorsque des parties d'applications seront requises, elles seront téléchargées sur le poste client. Dans un environnement diffusé, ce mécanisme de téléchargement devrait reposer, par exemple, sur une rediffusion cyclique de toutes les parties de l'application. Il est laissé à la responsabilité du client de posséder un environnement d'exécution qui sache interpréter les parties d'application, présente l'application à l'utilisateur et dirige l'interaction locale avec l'utilisateur.

Toute application MHEG-5 est composée de scènes et d'objets communs à toutes les scènes. Une scène contient un groupe d'objets utilisés pour présenter l'information (graphique, sonore, vidéo, etc.) avec des comportements localisés basés sur des amorçages d'événements (par exemple, pousser le bouton gauche déclenche un son). Une scène au plus est active à un moment donné. La navigation dans une application est effectuée grâce à des transitions entre scènes.

Le système interactif a la possibilité d'afficher des objets visuels dans un système de coordonnées rectangulaires avec une taille fixe, et de jouer des objets audibles. Les dispositifs d'entrée de l'utilisateur, à savoir la télécommande, le contrôleur de jeu, etc., peuvent être utilisés avec l'exécutable pour permettre les interactions avec les applications.

Les figures du présent paragraphe informatif présentent le diagramme de classes des classes objets définies par la présente Recommandation. La signification en est donnée dans la Figure 1.

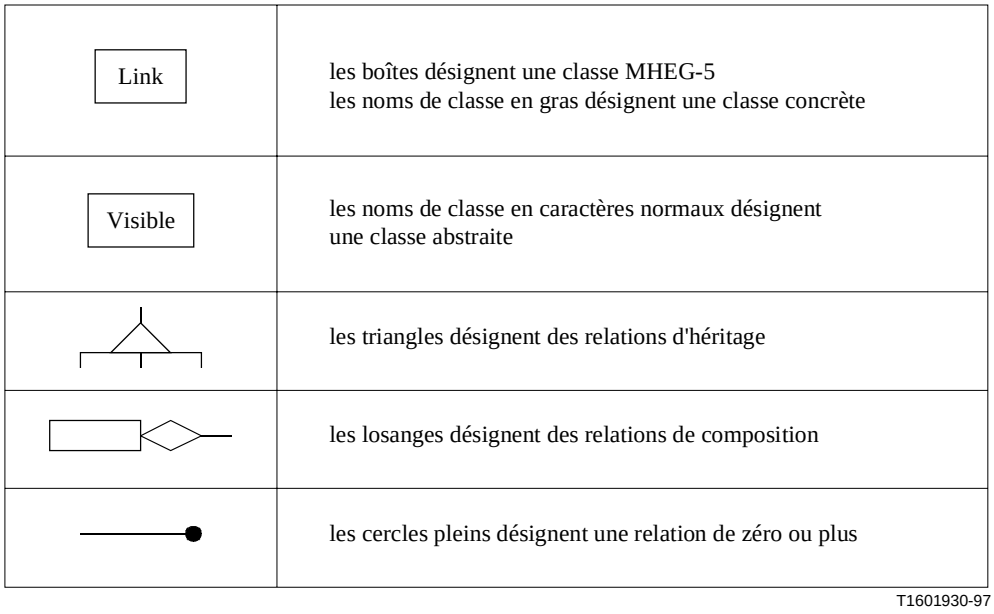


Figure 1/T.172 – Légende des diagrammes de classe

[1] Le serveur peut être un serveur *virtuel*, comme le regroupement d'un certain nombre de canaux de radiodiffusion sur un réseau de radiodiffusion.

La Figure 2 présente un aperçu général sur le niveau le plus haut de la hiérarchie des classes MHEG-5. La partie suivante du présent paragraphe introduit les concepts définis dans la présente Recommandation en expliquant les classes décrites dans la Figure 2 ainsi que leurs sous-classes.

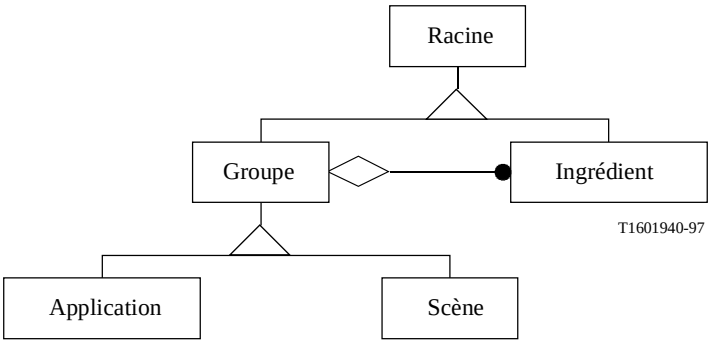


Figure 2/T.172 – Diagramme de la hiérarchie du sommet de la classe

5.1 Root *(Racine)*

C'est la classe de base abstraite pour toute les autres classes MHEG-5[2]. Ses fonctionnalités principales sont de fournir la sémantique de comportement générique MHEG-5 (activation, désactivation, préparation, destruction), et de fournir un mécanisme pour identifier les objets.

5.2 Group *(Groupe)*

C'est une classe de base abstraite pour les classes Application et Scène. Ses fonctionnalités principales sont d'autoriser le regroupement d'objets d'autres classes en vue d'un échange entre des moteurs MHEG-5 et d'autres entités (semblable à un "set" dans la terminologie orientée-objet normalisée). Les objets regroupés par cette classe sont des objets de la classe Ingrédient. Chaque Ingrédient est toujours contenu dans un seul Groupe. Les objets à l'intérieur d'un Groupe peuvent être référencés à partir d'autres objets dans d'autres Groupes sous certaines conditions (voir ci-dessous).

5.3 Application *(Application)*

Les objets de la classe Application regroupent des objets de la classe Ingrédient. La classe Application possède aussi la contrainte sémantique de n'avoir qu'un seul objet Application actif[3] à la fois, et qu'aucun autre objet ne peut être actif tant qu'un objet Application est actif.

Un moteur MHEG-5 en situation d'attente commence une application en préparant et en activant l'objet Application correspondant. Lorsque l'objet Application devient actif, il exécute automatiquement une action *OnStartUp,* qui peut être utilisée pour exécuter le premier objet Scène de l'application. Si (exactement) un objet Application est actif lorsque n'importe quel autre objet est actif, les ingrédients contenus dans l'objet application sont rendus visibles et disponibles aux autres objets simultanément actifs. Plus spécialement, n'importe quels Ingrédients contenus dans un objet Application sont rendus disponibles pour la Scène active. Ceci peut être utilisé pour décrire un comportement valide sur l'application toute entière.

---

[2] A l'exception des classes mélangées abstraites et de la classe Action.

[3] Pour la définition du terme "actif", voir 3.3.

5.4 Scene *(Scène)*

Les objets de la classe Scène regroupent des objets de la classe Ingrédient. La finalité de la classe Scène est de permettre une présentation coordonnée spatiale et temporelle. Un seul objet Scène peut être actif à un instant donné à l'intérieur d'un moteur MHEG-5. Un objet Scène doit être actif pour afficher tout ingrédient qu'il soit contenu dans une Scène ou un objet Application.

La classe Scène fournit l'action spéciale *TransitionTo* qui permet d'effectuer une transition graphique entre deux scènes. Les objets contenus dans une scène ne peuvent être affichés que lorsque cette scène est active. Un objet ayant besoin d'être affiché dans plusieurs scènes (à savoir demandant une présentation ininterrompue au delà d'une transition de scène) doit être contenu dans un objet application. Enfin, la classe Scène fournit l'information sur le système de coordonnées à utiliser pour la présentation visuelle.

5.5 Ingredient (*Ingrédient)*

La classe Ingrédient est une classe de base abstraite pour les classes Lien, Programme, Palette, Police, FormeCurseur, Variable et Présentable. Les sous-classes de la classe Ingrédient sont présentées dans la Figure 3. La fonctionnalité principale de la classe Ingrédient est de spécifier le comportement générique des objets pouvant faire partie d'un objet Scène ou Application.

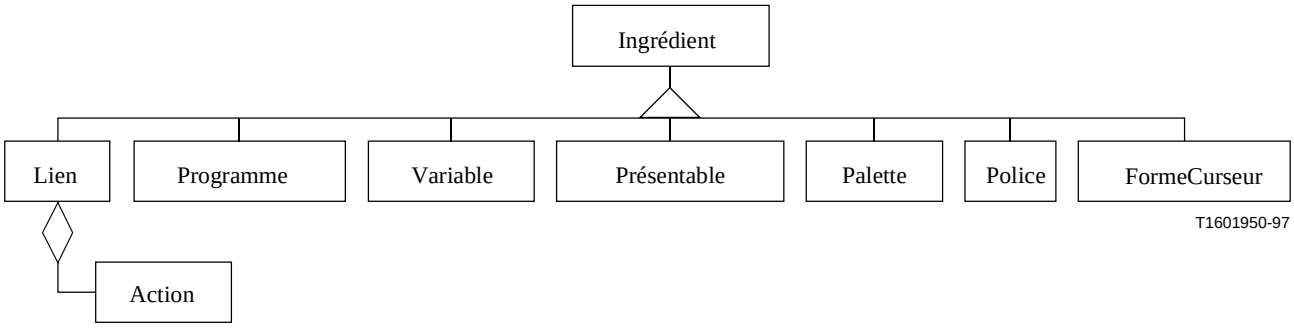


Figure 3/T.172 – Diagramme des sous-classes de la classe Ingrédient

5.6 Link *(Lien)*

Les objets Lien sont utilisés pour exprimer le comportement des applications MHEG-5. Un objet Lien comprend une condition et un objet Action. Lorsque la partie condition est évaluée à *True*, le lien est dit "à amorcer"; ceci conduit à l'exécution de l'objet Action associé. La condition contient trois parties: un code événement (identifiant l'événement sur lequel on doit réagir), une référence dont l'événement doit émaner, et une valeur spécifiant la valeur requise du paramètre d'événement. En d'autres mots, un Lien est amorcé seulement s'il est actif et seulement si l'événement approprié est généré par l'objet adéquat tout en contenant le paramètre d'événement approprié. Les Liens actifs feront partie d'une Scène active ou d'une Application active.

5.7 Action *(Action)*

Un objet Action possède la fonctionnalité d'exécuter, sous forme de séquence synchrone, une série d'actions élémentaires comme résultat de l'amorçage d'un lien. Une action élémentaire est composée de l'objet cible de l'action et d'une liste de valeurs représentant les paramètres de l'action. En fait, cibler une action élémentaire en direction d'un objet correspond à l'appel d'une méthode d'objet dans

tout langage de programmation orienté-objet ordinaire. Toutes les actions élémentaires disponibles sont listées au paragraphe 49.

La classe Action n'hérite d'aucune autre classe MHEG-5. Tout spécialement, elle n'hérite pas de la classe Racine, ce qui signifie que les objets Actions ne peuvent pas être adressées comme des entités propres.

5.8 Program *(Programme)*

La classe Programme fournit les fonctionnalités pour aller chercher un morceau de code procédural à partir de l'intérieur du contexte MHEG-5 et d'échanger des paramètres avec lui.

La classe Programme possède les trois sous-classes suivantes correspondant aux trois types d'appels de procédure pouvant être effectués:

- *ResidentProgram* (Programme Résident)

  appel procédural à un morceau de code spécifique au dispositif sur lequel tourne l'environnement MHEG-5. On peut l'utiliser pour appeler, par exemple, des bibliothèques temps réel spécifiques;

- *RemoteProgram* (Programme Distant)

  appel procédural à un morceau de code situé sur un dispositif différent de celui sur lequel tourne le moteur MHEG-5. On peut l'utiliser, par exemple, pour implémenter un appel de programme distant dont le corps de programme effectif est situé sur le serveur dans un système client-serveur;

- *InterchangedProgram* (Programme Echangé)

  appel procédural à un morceau de code échangé comme faisant partie d'un objet MHEG-5. La finalité de cette classe est de fournir les fonctionnalités requises pour échanger des morceaux de code procédural ainsi que pour les appeler.

5.9 Palette, Font, and CursorShape *(Palette, Police et FormeCurseur)*

La classe Palette fournit la possibilité d'encapsuler la représentation codée d'un tableau de correspondance de couleurs (CLUT, *colour look-up table*). Un tableau CLUT a pour fonction de traduire un index de couleur dans une valeur de couleur réelle. Une Palette peut être utilisée, par exemple, avec des phototrames pour spécifier les couleurs dans lesquelles la phototrame doit être visualisée.

Pareillement, la classe Police permet aux applications d'encapsuler la représentation d'une police. Un objet Police, associé à un objet Texte, est utilisé pour afficher le texte de cet objet.

La classe FormeCurseur permet aux applications d'encapsuler la représentation codée des phototrames, masques et de toutes autres données nécessaires à l'affichage d'un curseur de déplacement. La forme du curseur de déplacement peut être positionnée et ensuite modifiée en utilisant une méthode de la classe Scène.

Pour ce qui concerne les Polices, Palettes et FormeCurseurs, la représentation réelle des objets n'est pas spécifiée dans la présente Recommandation. Cependant, un domaine applicatif dans lequel les Polices et les CLUTs et les curseurs sont des attributs nécessaires aux applications peuvent spécifier leur codage et leur sémantique.

5.10 Variable *(Variable)*

La classe Variable fournit la possibilité de stocker et de récupérer des valeurs. La classe Variable possède cinq sous-classes correspondants à cinq types de variables différents.

- BooleanVariable *(Booléen)*;
- IntegerVariable *(Entier)*;
- OctetStringVariable *(Chaîne d'octets)*;
- ObjectRefVariable *(Référence à objet)*;
- ContentRefVariable *(Référence à contenu).*

Les utilisations possibles de Variables comprennent le passage de paramètres en direction de ou en retour d'appels de Programme, le stockage de l'état des autres objets MHEG-5, le passage indirect de valeurs de paramètre à des actions et les indirections d'adresse (identiques à des pointeurs sur des objets MHEG-5).

5.11 Presentable *(Présentable)*

La classe Présentable est une classe abstraite de base pour les classes MHEG-5 suivantes: Audio, Visible, GroupeJeton et Flux. Les sous-classes de la classe Présentable sont présentées dans la Figure 4.

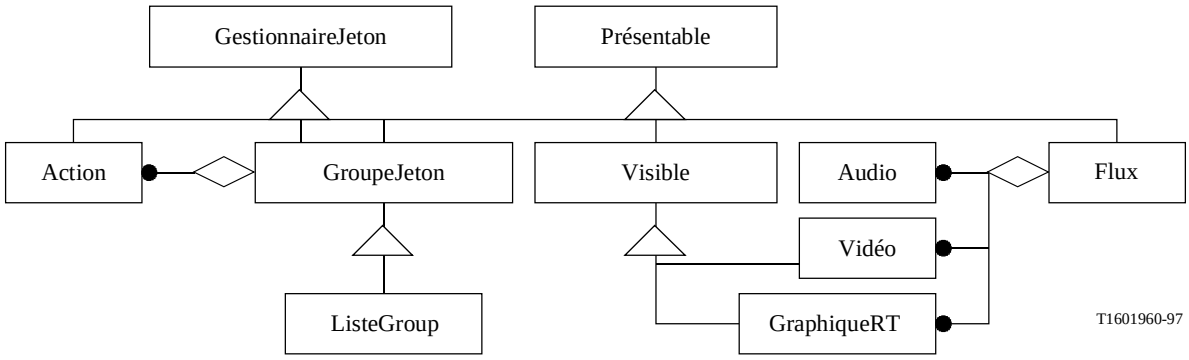


Figure 4/T.172 – Diagramme des sous-classes de la classe Présentable

Les objets de la classe Présentable représentent l'information qui peut directement être vue ou entendue par l'utilisateur. La manipulation de la représentation codée réelle des données *Content* est une fonctionnalité importante de la classe Présentable. Ceci peut être fait soit par inclusion ou par référence. Dans le premier cas, les données *Content* sont réellement transmises comme faisant partie intégrante de l'objet Présentable tandis que dans le cas suivant, l'objet Présentable fournit seulement une référence externe aux données.

5.12 TokenGroup *(GroupeJeton)*

La classe GroupeJeton fournit un service pour déplacer un jeton logique parmi un ensemble d'objets Visible. Cette structure peut être utilisée, par exemple, pour gérer les changements de mise en visibilité parmi un ensemble de boutons ou d'autres éléments d'une Scène. Des ensembles d'actions peuvent aussi être attachées aux objets et exécutées à la demande sur l'objet Visible possédant le jeton. Le dernier attribut fournit un moyen compact d'exprimer un comportement lorsqu'un élément du groupe obtient ou perd la mise en visibilité.

5.13 ListGroup *(ListeGroupe)*

La classe ListeGroupe complète la classe GroupeJeton en fournissant des fonctionnalités de sélection des objets dans une longue liste. Sa meilleur utilisation est pour l'implémentation d'un menu de sélection, d'un groupe de cases à cocher, d'un carrousel de phototrames, d'un formulaire, d'une liste d'éléments scrollables, etc. De plus, la classe ListeGroupe fournit des services d'ajout et de suppression dynamique d'éléments au groupe.

5.14 Stream *(Flux)*

La classe Flux définit un multiplex de média continus synchronisés dans le temps. Des objets Audio, Vidéo et GraphicsRT peuvent être des flux élémentaires d'un multiplex Flux: ils ont pour objectif d'être présentés simultanément à l'utilisateur. Cette structure peut être utilisée, par exemple, pour présenter une vidéo et un son de manière synchronisée et pour commuter d'un canal son à un autre. Les données *Content* de l'objet Flux sont une référence à un multiplex réel contenant les flux élémentaires et les données nécessaires à leur synchronisation. Pendant la phase de synchronisation, le gestionnaire de flux génère des événements basés sur le temps ou de type marqueur pouvant être utilisés par l'application MHEG-5 pour amorcer des Liens.

5.15 Audio *(Audio)*

La classe Audio implémente une séquence de donnés audio pouvant être utilisée comme flux élémentaire d'un multiplex Flux

5.16 Interactible

La classe Interactible est une classe transverse abstraite héritée par les classes MHEG-5 HyperTexte, ChampEntrée, Curseur et Bouton. Ses fonctionnalités principales sont de permettre à l'utilisateur d'interagir avec les objets de ses sous-classes. Ces interactions autorisent l'utilisateur à changer l'état et/ou l'apparence des objets en entrant, par exemple, du texte dans un objet ChampEntrée. Lorsqu'une interaction intervient, une certaine classe d'événements, appelée les "événements *UserInput*" ne sont pas visibles des objets Liens, puisqu'on suppose que ces événements sont utilisés pour l'interaction de l'utilisateur. L'interaction peut être interrompue en adressant une action d'un certain type vers l'objet Interactible.

L'autre fonctionnalité de la classe Interactible est la possibilité de générer des événements associés aux curseurs de déplacement (*CursorEnter*, *CursorLeave*).

5.17 Visible *(Visible)*

La classe Visible implémente les fonctionnalités associées à la présentation des morceaux de contenus visuels sur l'écran d'affichage. Les sous-classes de la classe Visible sont présentées dans la Figure 5.

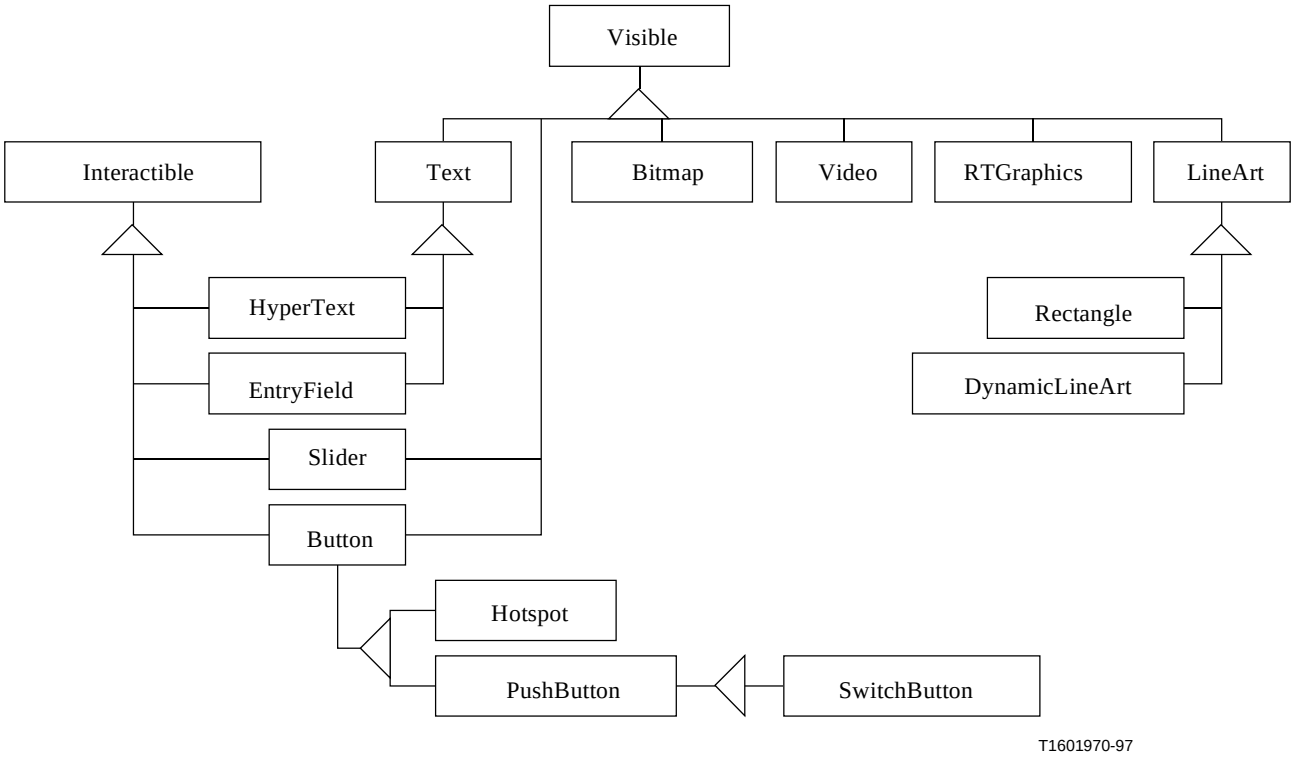


Figure 5/T.172 – Diagramme des sous-classes de la classe Visible

Les sous-classes de la classe Visible sont décrits ci-dessous:

- *LineArt*

  Un objet de la classe *LineArt* représente un objet graphique ayant une représentation vectorielle. On peut l'utiliser, par exemple, pour présenter des objets poly-traits, ellipses, courbes de bézier, etc.;

- *DynamicLineArt*

  Un objet de la classe *DynamicLineArt* représente un objet graphique pouvant être modifié dynamiquement. On peut l'utiliser pour tracer des traits ou des courbes devant être présentées au fil de l'eau;

- *Rectangle*;

- *Phototrame*;

- *Vidéo*;

- *RTGraphics*

  Un objet de la classe RTGraphics (Graphique Temps Réel) représente un flux d'objets graphiques affichés selon une synchronisation et une localisation autonome. Le flux RTGraphics peut être utilisé, par exemple, en synchronisation avec une vidéo et un son pour créer une application de sous-titrage;

- *Text*

  Les objets Texte représentent des chaînes de texte. Un objet Texte peut être associé avec un objet Police qui décrit la police dans laquelle le texte doit être affiché (si aucune police n'est mentionnée, la police par défaut est utilisée). Le Texte a deux sous-classes, Hypertext et ChampEntrée, chacune implémentant des types de textes interactifs différents;

- *Slider* (Curseur)

  Un curseur est un Interactible permettant à l'utilisateur d'afficher une position de manière linéaire à l'intérieur d'un intervalle donné (identifié par des valeurs minimales et maximales);

- *Button* (Bouton)

  La classe Bouton possède deux sous-classes: *PushButton* et *HotSpot*. La classe *PushButton* possède une sous-classe appelée *SwitchButton*. Les boutons sont des zones rectangulaires avec lesquelles l'utilisateur interagit. Aux événements générés sur chacun des trois types de boutons sont associés des comportements différents. Les *PushButtons* et les *SwitchButtons* sont associés à un élément texte représentant les textes à afficher au milieu du bouton.

6 Structure de la présente Recommandation

Les paragraphes suivants de la présente Recommandation définissent la sémantique des classes MHEG-5. On utilise pour cela une notation de syntaxe abstraite pour décrire les attributs des objets échangés. Les sémantiques des classes d'objets sont décrites dans un texte normatif. Les paragraphes 51 à 54 définissent des mécanismes normatifs qu'il est demandé à un moteur MHEG-5 d'implémenter. L'Annexe A définit la syntaxe de la forme définitive devant être utilisée pour tout échange d'objet. L'Annexe B définit un format d'échange textuel qui établit un mappage biunivoque avec le format binaire d'échange de la forme définitive.

7 Notations

Les notations suivantes sont utilisées dans tous les paragraphes suivants pour décrire les classes MHEG-5 définies dans la présente Recommandation.

<Nom de la classe>

Description	<Brève description de la sémantique de la classe>
classe de base	<Nom de la classe de base>
Sous-classes	<Liste des sous-classes, s'il y en a>
Etat	<classe abstraite | classe concrète>

7.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de la classe.

7.1.1 Attributs hérités

Besoins et contraintes sur les attributs hérités des classes de base.

Nom d'attribut	Défini dans	Besoins et contraintes
< Nom d'attribut >	<Nom de classe>	<Contraintes spécifiques de la sous-classe>

7.1.2 Attributs échangés propres

Liste des attributs échangés pour cette classe.

<Nom d'attribut> <Description de l'attribut>

<Type de l'attribut>

<Valeur par défaut>

Lorsque l'attribut est optionnel, la valeur par défaut est la valeur à utiliser quand l'attribut n'est pas codé. Lorsque l'attribut est obligatoire, la valeur par défaut est un choix parmi les valeurs les plus habituellement utilisées.

7.1.3 Attributs internes propres

Liste des attributs internes pour cette classe.

<*Nom d'attribut*> <Description de l'attribut>

<Valeur initiale>

7.2 Evénements

Ce sont les événements pouvant être générés à partir des objets de cette classe. Ils sont utilisés pour exprimer des conditions de liens qui seront vérifiées lorsqu'un tel événement est généré.

<*Nom d'événement*> <Description de l'événement>

<Contexte d'apparition de l'événement>

<Lorsqu'elles existent, description des données d'événements associées à l'événement>

7.3 Comportement interne

Les comportements internes suivants sont définis pour la plupart des classes MHEG-5:

- *Preparation* (Préparation);
- *Destruction* (Destruction);
- *Activation* (Activation);
- *Deactivation* (Désactivation).

En plus, un comportement interne *Interaction* est défini pour quelques classes MHEG-5. Ces comportements correspondent à des besoins sémantiques d'opérations internes au moteur MHEG-5, semblables à la notion de méthodes privées de classe dans la terminologie orientée objet. Ils ne sont ni codés, ni transmis à l'intérieur d'une description d'Application.

Quand un comportement n'est pas décrit pour une classe MHEG-5 donnée, on y applique la sémantique du comportement de sa classe de base.

<*Nom du comportement*> <Sémantique du comportement à l'intérieur du contexte de la classe courante>

7.4 Effet des actions MHEG-5

Définit la sémantique et la syntaxe des actions MHEG-5 pouvant être adressées à la classe MHEG-5 courante.

<Nom d'action> <Sémantique de l'action à l'intérieur du contexte de la classe courante>

7.5 Description formelle

Description d'un objet MHEG-5 codé de la classe courante en notation (EBNF, *extended Backus-Naur form*):

```
Nom de classe          -->  Première partie,
                            Seconde partie
Première partie        -->  Sous partie | Alternative
Seconde partie         -->  Terminal (constante ou type)
```

8 Classe Root *(Racine)*

Description: classe Racine des classes MHEG-5

classe de base: sans objet

Sous-classe: *Group, Ingredient*

Etat: classe abstraite

8.1 Attributs

Le présent sous-paragraphe définit les attributs internes, échangés et hérités de cette classe.

8.1.1 Attributs hérités

Cette classe n'a pas d'attributs hérités.

8.1.2 Attributs échangés propres

Cette classe définit les attributs échangés supplémentaires suivants:

ObjectIdentifier (Identificateur d'objet)

C'est une structure de données obligatoire comprenant les parties suivantes:

- *GroupIdentifier* (identificateur de groupe*)* Optionnel.

  Identificateur unique de groupe d'objets MHEG-5.

  – *OctetString* Optionnel.

  – *Valeur* par défaut: le *GroupIdentifier* de l'objet *Group* à l'intérieur duquel l'objet est codé.

  La structure réelle de ce paramètre n'est pas définie dans la présente Recommandation. Cependant, le domaine applicatif définit une telle structure. Voir le paragraphe 51.

- *ObjectNumber* (numéro d'objet).

  Identificateur unique de tout objet MHEG-5 à l'intérieur d'un objet *Group*.

  – Entier.

8.1.3 Attributs MHEG-5 internes propres

Cette classe définit les attributs internes suivants:

AvailabilityStatus (Etat de disponibilité*)*

Etat de disponibilité de l'objet.

Lorsque l'état *AvailabilityStatus* de cet objet est *True*, l'objet est disponible; ceci signifie que le comportement *Preparation* de l'objet s'est terminé avec succès.

Lorsque l'état *AvailabilityStatus* de cet objet est *False*, l'objet n'est pas disponible; ceci signifie que le comportement *Preparation* ne s'est pas terminé avec succès ou n'a pas été invoqué, ou encore que le comportement *Destruction* a été appliqué avec succès.

- valeur *Boolean*.
- Valeur par défaut: *False*.

RunningStatus (Etat en cours)

Etat d'activité de l'objet.

Lorsque l'état *RunningStatus* de cet objet est *True*, l'objet est actif; ceci signifie que le comportement *Activation* de l'objet s'est terminé avec succès.

Lorsque l'état *RunningStatus* de cet objet est *False*, l'objet est inactif; ceci signifie que le comportement *Activation* ne s'est pas terminé avec succès ou n'a pas été invoqué, ou encore que le comportement *Deactivation* a été appliqué avec succès.

- valeur *Boolean*.
- Valeur par défaut: *False*.

8.2 Evénements

Cette classe définit les événements suivants:

IsAvailable (EstDisponible)

Cet événement est généré lorsque l'attribut *AvailabilityStatus* passe de *False* à *True* comme résultat du comportement *Preparation* s'étant terminé avec succès pour cet objet.

NOTE – Cet événement a pour but d'indiquer au moteur MHEG-5 que l'objet est disponible et peut être activé.

- Aucune donnée associée.

ContentAvailable (ContenuDisponible)

Cet événement est généré lorsque l'objet et son contenu sont disponibles et dans un état optimisé pour le moteur MHEG-5. L'objectif de cet attribut est d'indiquer au moteur MHEG-5 que le comportement *Activation* peut se produire dans le temps. Cet événement est généré de manière asynchrone grâce au comportement *Preparation* appliqué sur cet objet.

NOTE – Chaque moteur MHEG-5 peut choisir de donner un sens différent à ce niveau de disponibilité de contenu. La présente Recommandation ne spécifie aucune signification ou contrainte de temps à ce sujet.

- Aucune donnée associée.

IsDeleted
(EstDétruit)

Cet événement est généré lorsque l'attribut *AvailabilityStatus* passe de *False* à *True* comme résultat du comportement *Destruction* s'étant terminé avec succès pour cet objet.

- Aucune donnée associée.

IsRunning
(EstEnCours)

Cet événement est généré lorsque l'attribut *RunningStatus* passe de *False* à *True* comme résultat du comportement *Activation* s'étant terminé avec succès pour cet objet.

- Aucune donnée associée.

IsStopped
(EstArrêté)

Cet événement est généré lorsque l'attribut *RunningStatus* passe de *False* à *True* comme résultat du comportement *Deactivation* s'étant terminé avec succès pour cet objet.

- Aucune donnée associée.

8.3 Comportements internes

Cette classe définit les comportements internes suivants:

Preparation
(Préparation)

Ce comportement a pour sémantique de base d'allouer toutes les ressources requises dans le but de manipuler ou de présenter cet objet.

S'applique la suite des actions suivantes:

1) si l'attribut *AvailabilityStatus* de l'objet vaut *True*, annuler le comportement. Autrement:
2) récupérer l'objet d'une entité à l'extérieur du moteur;
3) affecter sa valeur initiale à chaque attribut interne de l'objet;
4) affecter la valeur *True* à l'attribut *AvailabilityStatus*;
5) générer un événement *IsAvailable*.

Les étapes ci-dessus sont exécutées de manière *synchrone*. L'étape suivante est asynchrone.

6) générer un événement *ContentAvailable*.

Destruction
(Destruction)

Ce comportement a pour sémantique de base de demander au moteur MHEG-5 la destruction de l'objet.

S'applique la suite des actions suivantes:

1) si l'attribut *AvailabilityStatus* de cet objet vaut *False*, annuler le comportement. Sinon:
2) si l'attribut *RunningStatus* de cet objet vaut *True*:
   a) appliquer le comportement *Deactivation*;
   b) attendre un événement *IsStopped* de l'objet.

   Ceci sera fait de manière synchrone.

3) si l'attribut *RunningStatus* de l'objet vaut *False*, exécuter les actions suivantes de manière synchrone;

4) Si l'attribut *GroupCachePriority* de l'objet lui-même ou du groupe auquel appartient l'objet vaut la valeur 0, le moteur MHEG-5 libérera toutes les ressources allouées à l'objet.

Il faut noter que *GroupCachePriority* est défini dans la classe *Group*;

5) si l'attribut *GroupCachePriority* de l'objet lui-même ou du groupe auquel appartient l'objet est différent de 0, le moteur MHEG-5 peut décider de soit réellement libérer les ressources allouées à cet objet ou de les cacher;

6) générer un événement *IsDeleted*;

il faut noter que l'événement *IsDeleted* sera généré que les ressources allouées ci-dessus soient réellement libérées ou non. L'objet est détruit dans le sens défini par la présente Recommandation, même dans le cas où des ressources associées ne le sont pas.

Activation

(Activation)

Ce comportement a pour sémantique de base de rendre immédiatement cet objet actif.

S'applique la suite des actions suivantes:

1) si l'attribut *RunningStatus* de cet objet vaut *True*, annuler le comportement. Sinon:

2) si l'attribut *AvailabilityStatus* de cet objet vaut *False*:

a) appliquer le comportement *Preparation* à cet objet;

b) attendre l'événement *IsAvailable* de cet objet.

Ces étapes sont exécutées de manière synchrone, c'est-à-dire que le moteur n'effectuera pas d'autres actions tant que le comportement *Preparation* n'est pas terminé.

NOTE 1 – L'effet du comportement *Activation* (par exemple l'affichage d'une phototrame) se poursuivra même après la terminaison du comportement lui-même.

NOTE 2 – La génération de l'événement *IsRunning* et la modification de l'attribut interne *RunningStatus* font partie du comportement *Activation* des sous-classes de la classe *Root*.

Deactivation

(Désactivation)

Ce comportement a pour sémantique de base de notifier au moteur MHEG-5 de désactiver immédiatement cet objet.

S'applique la suite des actions suivantes:

1) si l'attribut *RunningStatus* de cet objet vaut *False*, annuler le comportement. Sinon:

2) mettre l'attribut *RunningStatus* de l'objet à *False*;

3) générer l'événement *IsStopped*.

8.4 Effet des actions MHEG

Cette classe définit les actions MHEG-5 applicables suivantes:

GetAvailability-Status (*AvailabilityStatusVar*)

Récupère l'état de disponibilité (Variable de statut de disponibilité)

Affecte à la variable *AvailabilityStatusVar* la valeur de l'attribut *AvailabilityStatus*.

NOTE – Une action *GetAvailabilityStatus* adressée à un objet inexistant dans le moteur MHEG n'est pas une erreur, le résultat est *False*.

Disposition pour l'utilisation:

- *AvailabilityStatusVar* fera référence à un objet *BooleanVariable*.

Description de la syntaxe:

```
GetAvailabilityStatus      -->  Target,
                                AvailabilityStatusVar
Target                     -->  GenericObjectReference
AvailabilityStatusVar      -->  ObjectReference
```

GetRunningStatus (*RunningStatusVar*)

Récupère l'état en cours (Variable état en cours)

Met la variable référencée par *RunningStatusVar* à la valeur de l'attribut *RunningStatus*.

Dispositions pour l'utilisation:

- l'objet *Target* sera disponible;
- *RunningStatusVar* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
GetRunningStatus           -->  Target,
                                RunningStatusVar
Target                     -->  GenericObjectReference
RunningStatusVar           -->  ObjectReference
```

8.5 Description formelle

```
Root Class              -->  ObjectIdentifier
ObjectIdentifier        -->  ObjectReference
```

9 Classe *Group*

Description: définit la structure et le comportement des objets utilisés comme composition d'*Ingredients*

classe de base: *Root*

Sous-classes: *Scene*, *Application*

Etat: classe abstraite

9.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

9.1.1 Attributs hérités

Cette classe possède les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ObjectIdentifier (Identificateur d'objet)	*Root*	Cet attribut est obligatoire pour cette classe. La partie GroupIdentifier de cet attribut est obligatoire et sera unique à l'intérieur de l'espace de nom du domaine applicatif. La partie ObjectNumber de cet attribut sera mise à 0.

9.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

StandardIdentifier
(Identificateur de Standard)

C'est une séquence optionnelle comprenant deux entiers. S'il est codé:

- le premier entier vaudra toujours 2, signifiant "Joint ISO ITU";
- le second entier vaudra toujours 19, signifiant MHEG.

StandardVersion
(Version de Standard)

C'est un entier simple optionnel définissant la version de la présente Recommandation à laquelle les objets *Group* et tous leurs éléments se conforment.

- Entier optionnel. Si codé, sa valeur est mise à 1.
- Valeur par défaut: 1.

ObjectInformation
(Information sur Objet)

C'est un *OctetString* optionnel. S'il est codé, il contient de l'information sur les objets codés de ce *Group*. Une telle information peut comprendre de l'information sur les objets *Name*, *Owner*, *Version*, *Date*, *Keywords*, *Copyright*, *License*, et *Comments*.

OnStartUp
(Lors du lancement)

Ensemble d'actions élémentaires à exécuter à la fin du comportement *Activation* appliqué à l'objet *Group*.

- Insertion optionnelle d'un objet *Action*.
- Valeur par défaut: aucune.

OnCloseDown
(Lors de la fermeture)

Ensemble d'actions élémentaires à exécuter à la fin du comportement *Deactivation* appliqué à l'objet *Group*.

- Insertion optionnelle d'un objet *Action*.
- Valeur par défaut: aucune.

OriginalGroup-CachePriority
(Priorité de cache de Groupe d'origine)

Alerte à destination du moteur MHEG-5 concernant le degré de pertinence pour cacher ce *Group* et ses *Ingredients* lorsque ceux-ci sont détruits.

Valeur de *GroupCachePriority* lorsque l'objet *Group* est préparé.

- Entier optionnel dans l'intervalle [0, 255].
- Valeur par défaut: 127.
- Valeur spécifique à 0 signifiant que le cache n'est pas autorisé pour ce *Group* et ses *Ingredients*.

NOTE – Comme il est spécifié au 4.2.2, le fait de pouvoir cacher n'importe quel type est un attribut optionnel du moteur MHEG-5.

Items

(éléments)

Ensemble des objets *Ingredient* appartenant à l'objet *Group*. Lorsque le *Group* ne contiendra aucun *Ingredient*, cet attribut ne sera pas codé. Lorsque cet attribut sera codé, il contiendra au moins un *Ingredient*.

- Attribut optionnel.
- Séquence d'insertion d'objets *Ingredient*.
- Valeur par défaut: sans objet.

9.1.3 Attributs internes propres

Cette classe définit l'attribut interne suivant:

GroupCachePriority

(Priorité de cache de Groupe)

Alerte à destination du moteur MHEG-5 concernant le degré de pertinence pour cacher ce *Group* et ses *Ingredients* lorsque celui-ci est détruit.

La priorité *GroupCachePriority* peut être comparée avec celle d'autres objets *Group* pour déterminer le numéro de groupe ayant la plus haute probabilité d'être à nouveau demandé par l'application, une fois qu'il a été détruit. Plus la valeur est élevée et plus le niveau de priorité est élevé. Il est laissé à la responsabilité du développeur d'application de gérer ces numéros dans un intervalle cohérent. Il est recommandé au moteur MHEG-5 de cacher des objets selon une préférence allant de la priorité la plus haute à la plus basse.

- Entier optionnel dans l'intervalle [0, 255].
- Valeur initiale: valeur de l'attribut *OriginalGroupCachePriority*.
- Valeur spécifique à 0 signifiant que le cache n'est pas autorisé pour ce *Group* et ses *Ingredients.*

9.2 Evénements

Cette classe possède les mêmes événements que ceux de sa classe de base, avec une sémantique identique.

9.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à la classe de base de cet objet:

Preparation

(Préparation)

1) Appliquer le comportement *Preparation* à tous les *Ingredients* de *Group* ayant l'attribut *InitiallyActive* mis à *True* et à tous les *Programs* de *Group* ayant l'attribut *InitiallyAvailable* mis à *True* selon leur ordre de rangement dans l'attribut *Items*.

2) Appliquer le comportement *Preparation* comme s'il était hérité de la classe de base.

Destruction

(Destruction)

1) Appliquer le comportement *Destruction* à tous les *Ingredients* de *Group* dans l'ordre inverse de leur apparition dans l'attribut *Items*.
2) Appliquer le comportement *Destruction* comme hérité de la classe de base.

Activation

(Activation)

1) Appliquer le comportement *Activation* comme hérité de la classe de base.
2) Exécuter l'action contenue dans l'attribut *OnStartUp*.
3) Appliquer le comportement *Activation* à tous les *Ingredients* du *Group* ayant l'attribut *InitiallyActive* mis à *True* dans leur ordre d'apparition dans la liste *Items*.
4) Mettre l'attribut *RunningStatus* de l'objet Group à *True*.
5) Générer l'événement *IsRunning*.

Deactivation

(Désactivation)

Si le *Group* n'est pas actif, ignorer le comportement. Si le *Group* est actif, effectuer les trois étapes suivantes:

1) Exécuter l'action contenue dans l'attribut *OnCloseDown*.
2) Appliquer le comportement *Deactivation* à tous les *Ingredients* actifs du *Group*, dans leur ordre inverse d'apparition dans l'attribut *Items*.
3) Appliquer le comportement *Deactivation* comme hérité de la classe de base.

9.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique. Il faut y ajouter l'action suivante:

SetCachePriority (*NewCachePriority*)

Affecte la priorité de cache

(Nouvelle priorité de cache)

Met l'attribut *GroupCachePriority* à la valeur *NewCachePriority*.

Dispositions pour l'utilisation:

- l'objet *Target* sera disponible;
- *NewCachePriority* sera affecté à l'intérieur de l'intervalle [0, 255].

Description de la syntaxe:

```
SetCachePriority          -->   Target,
                                NewCachePriority
Target                    -->   GenericObjectReference
NewCachePriority          -->   GenericInteger□
```

9.5 Description formelle

```
Group Class                        -->  Root Class,
                                        StandardIdentifier?,
                                        StandardVersion?,
                                        ObjectInformation?,
                                        OnStartUp?,
                                        OnCloseDown?,
                                        OriginalGroupCachePriority?,
                                        Items?
StandardIdentifier                 -->  Joint ISO ITU (2), MHEG (19)
StandardVersion                    -->  INTEGER
ObjectInformation                  -->  OctetString□
OnStartUp                          -->  Action Class
OnCloseDown                        -->  Action Class
OriginalGroupCachePriority         -->  INTEGER
Items                              -->  Item+
Item                               -->  ResidentProgram Class |
                                        RemoteProgram Class |
                                        InterchangedProgram Class |
                                        Palette Class |
                                        Font Class |
                                        CursorShape Class |
                                        BooleanVariable Class |
                                        IntegerVariable Class |
                                        OctetStringVariable Class |
                                        ObjectRefVariable Class |
                                        ContentRefVariable Class |
                                        Link Class |
                                        Stream Class |
                                        Bitmap Class |
                                        LineArt Class |
                                        DynamicLineArt Class |
                                        Rectangle Class |
                                        Hotspot Class |
                                        SwitchButton Class |
                                        PushButton Class |
                                        Text Class |
                                        EntryField Class |
                                        HyperText Class |
                                        Slider Class |
                                        TokenGroup Class |
                                        ListGroup Class
```

10 Classe Application

Description:	définit un ensemble d'objets *Ingredient* partagés à l'intérieur d'un domaine applicatif.
classe de base:	*Group*
Sous-classes:	sans objet
Etat:	classe concrète

10.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

10.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec une sémantique identique.

10.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

OnSpawnCloseDown

(Fermeture sur génération)

Objet *Action* à exécuter lorsque *l'Application* est fermée par l'ouverture d'une autre *Application* au moyen de l'action *Spawn*. Cela peut par exemple s'avérer utile pour stocker de l'information à utiliser lors du redémarrage de cette *Application*.

OnSpawnCloseDown peut seulement être invoquée par l'action *Spawn* (voir *Spawn*).

- Insertion optionnelle d'un objet *Action*.
- Valeur par défaut: aucune.

OnRestart

(Sur redémarrage)

Objet action à exécuter lorsque *l'Application* est relancée. Ceci peut être utile par exemple pour récupérer de l'information ayant été stockée par l'action *OnSpawnCloseDown*.

OnRestart ne peut être invoquée que par l'action *Quit* (voir *Quit*).

- Insertion optionnelle d'un objet *Action*.
- Valeur par défaut: aucune.

DefaultAttributes

(Attributs par défaut)

Définit des attributs par défaut ayant une portée sur toute l'application qui seront utilisés comme valeurs par défaut lorsqu'un attribut d'un *Ingredient* correspondant ne sera pas spécifié. Les attributs par défaut suivants peuvent être affectés:

CharacterSet

(Jeu de caractères)

Jeu de caractères par défaut, ou ensemble de jeux de caractères, pour afficher le texte dans toute l'application, sauf en cas de spécification contraire. Cet Entier sera codé avec une valeur représentant le jeu. Le domaine applicatif définira l'intervalle du jeu *CharacterSet* et sa sémantique.

NOTE – L'attribut *CharacterSet* d'*Application* fournit le jeu de caractères initial pour tous les objets contenant du texte en l'absence de spécification par *Application*.

- Entier optionnel.
- Valeur par défaut: sans objet.

BackgroundColour

(Couleur de fond)

Couleur par défaut à utiliser pour afficher le fond d'un objet *Text*. Cet attribut est interprété comme un indice basé zéro dans la table de correspondance de couleurs définie par l'attribut *PaletteRef*, ou comme une valeur directe de couleur; ceci dépendra du type d'attribut.

- *OctetString* ou *Entier* optionnel. Un *Entier* sera interprété comme un indice dans une *Palette*, un *OctetString* comme une valeur de couleur directe.
- Valeur par défaut: transparent.

TextColour

(Couleur de texte)

Couleur par défaut à utiliser pour afficher la couleur d'un objet *Text*. Cet attribut est interprété comme un indice basé zéro dans la table de correspondance de couleurs définie par l'attribut *PaletteRef*, ou comme une valeur directe de couleur; ceci dépendra du type d'attribut.

- *OctetString* ou *Entier* optionnel. Un *Entier* sera interprété comme un indice dans une *Palette*, un *OctetString* comme une valeur de couleur directe.
- Valeur par défaut: toute couleur.

Font

(Police)

Police par défaut à utiliser pour la présentation d'un objet *Text*.

L'attribut *Font* représente soit un nom de police (qui est résident dans le moteur MHEG-5) ou une référence à un objet *Font*.

Lorsque aucune référence à police n'est codée dans *Application*, l'objet *Text* est présenté en utilisant une police par défaut du moteur MHEG-5.

- Attribut optionnel.
- *OctetString* représentant un *FontName*, ou une référence à un objet *Font*.
- Valeur par défaut: police par défaut.

FontAttributes

(Attributs de police)

Les attributs de police par défaut tels que le style, la taille de caractère, la couleur de texte et la couleur de fond.

Le format de codage exact de l'attribut *FontAttributes* est en relation avec la valeur du type de l'objet *Font* mentionné par l'attribut *Font*.

- *OctetString* optionnel.
- Valeur par défaut: pas d'attribut spécifique.

BitmapContentHook

(Crochet de contenu de phototrame)

Valeur de *hook* par défaut pour tous les objets *Bitmap*.

- Entier optionnel.
- Valeur par défaut: sans objet.

StreamContentHook

(Crochet de contenu de flux)

Valeur de *hook* par défaut pour tous les objets *Stream*.

- Entier optionnel.
- Valeur par défaut: sans objet.

TextContentHook

(Crochet de contenu de texte)

Valeur de *hook* par défaut pour tous les objets *Text*.

- Entier optionnel.
- Valeur par défaut: Sans objet.

LineArtContentHook

(Crochet de contenu d'un graphique vectoriel lineArt)

Valeur de *hook* par défaut pour tous les objets *LineArt*.

- Entier optionnel.
- Valeur par défaut: sans objet.

Interchanged-ProgramContentHook

(Crochet de contenu de programme échangé)

Valeur de *hook* par défaut pour tous les objets *InterchangedProgram*.

- Entier optionnel.
- Valeur par défaut: sans objet.

ButtonRefColour

(Couleur de référence à bouton)

Couleur *ButtonColour* par défaut pour l'affichage de bouton.

- Entier optionnel ou *OctetString*.
- Valeur par défaut: sans objet.

HighlightRefColour

(Couleur de référence à surbrillance)

Couleur *HighlightRefColour* par défaut pour objet *Interactible*.

- Entier optionnel ou *OctetString*.
- Valeur par défaut: sans objet.

SliderRefColour

(Couleur de référence à curseur)

Couleur *SliderRefColour* par défaut pour objet *Slider*.

- Entier optionnel ou *OctetString*.
- Valeur par défaut: sans objet.

10.1.3 Attributs internes propres

Cette classe définit les attributs internes propres suivants:

LockCount

(Verrouillage de compte)

Spécifie si l'écran d'affichage est ou non figé.

Lorsque cet attribut est positif et différent de zéro, l'affichage ne répercutera aucun changement apporté aux Objets *Visibles* qui aurait dû avoir pour effet de modifier leur apparence. Cependant tous les changements seront répercutés en une seule fois dès que l'écran ne sera plus figé. Ceci est signalé au moyen de l'attribut *LockCount* mis dans ce cas à 0.

Les objets *Audio* (d'un multiplex *Stream*) continueront à être joués au travers d'une action *LockScreen:* on continuera à les entendre.

Les objets *Visibles* faisant partie d'un multiplex *Stream* continueront à être joués, mais tout autre changement sur ces objets (à savoir position, volume...) ne sera pas répercuté tant que l'écran restera figé. "Continue à être joué" signifie qu'ils continueront à déplacer leur attribut *CounterPosition* lorsque l'écran sera verrouillé, mais la mise à jour de la présentation graphique de ces objets pendant la phase de verrouillage de l'écran reste optionnelle. Sur certains moteurs, l'affichage physique continue à courir, sur d'autres l'image est figée jusqu'à ce que l'écran soit déverrouillé.

Lorsque l'écran est déverrouillé, l'affichage des objets Visibles sera compatible avec la valeur de l'attribut *CounterPosition*.

- Entier supérieur ou égal à 0.
- Valeur initiale: 0.

DisplayStack

(Pile d'affichage)

Liste ordonnée de références à objets *Visibles* indiquant l'organisation en niveaux graphiques des objets *Visibles* de l'application.

Les objets *Visibles* situés en bas de la pile *DisplayStack* seront affichés à l'arrière-plan de l'écran tandis que les objets *Visibles* en haut de cette pile seront affichés à l'avant-plan de l'écran.

Il faut noter que la pile *DisplayStack* peut contenir des références à des objets *Visibles* inactifs. Dans ce cas, les objets *Visibles* n'apparaissent tout simplement pas sur l'écran mais ils restent en tant qu'éléments valides de la pile *DisplayStack*.

- Liste ordonnée d'*ObjectReferences* à objets *Visibles*.
- Valeur initiale: liste vide.

10.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, l'événement suivant est défini:

EngineEvent

(Evénement moteur)

Cet événement est généré lorsqu'un événement particulier est apparu dans l'environnement du moteur MHEG-5. La présente Recommandation ne spécifie aucun des ces événements; le domaine applicatif peut spécifier la sémantique de chacun de ces événements et la valeur de leurs données associées.

- Données associées: *EventTag* (drapeau événement) – Entier.

10.3 Comportements internes

La sémantique du comportement interne suivant a été modifié par rapport à celui de la classe de base:

Deactivation

(Désactivation)

Exécute la séquence d'actions suivante:

1) Appliquer l'action *CloseConnection* à toutes les connexions auxiliaires ouvertes;

2) Appliquer le comportement *Deactivation* comme hérité de la classe de base.

10.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

StorePersistent (*StoreSucceeded*, *InVariables*, *OutFileName*)

(Stockage permanent, Stockage réussi, Variables d'entrée, Nom de fichier de sortie)

Demande au moteur MHEG-5 de sauvegarder la donnée d'une manière telle qu'elle puisse plus tard être récupérée par l'action *ReadPersistent*.

La donnée à sauvegarder est passée à travers un ensemble de variables référencées par le paramètre *InVariables*. Ces variables peuvent contenir des Booléens, Entiers, Chaînes d'octets, Références à objet et Références à contenu.

La donnée est sauvegardée dans une structure de donnée de fichier. Le nom *OutFileName* est un autre paramètre de l'action *StorePersistent*. La présente Recommandation ne définit pas les nature, structure, propriété, protection ou restriction de l'espace de fichier. Cependant les actions *StorePersistent* et *ReadPersistent* utiliseront le même espace de nom de fichiers.

L'effet de l'action *StorePersistent* est synchrone. Sur achèvement réussi de l'action *StorePersistent*, la variable référencée par *StoreSucceeded* sera mise à *True*, sinon à *False*.

Exemple: considérons l'action élémentaire suivante comprise dans l'attribut *OnSpawnCloseDown* d'une Application:

:StorePersistent (("myApp" 0) ("myApp" 1) (("scene1" 1) ("scene1" 2)) "myfile.txt"). ("myApp" 0) est l'identificateur *ObjectIdentifier* de l'Application courante. ("scene1" 1) et ("scene1" 2) sont les identificateurs *ObjectIdentifiers* de variables détenant une information relative à l'application courante, à savoir de l'information utilisateur. Ces données seront stockées dans un fichier appelé "myfile.txt". La variable ("myApp" 1) fait référence à un Booléen indiquant si l'action élémentaire à réussi ou non.

Les données peuvent être récupérées au début ou au redémarrage de cette Application (ou d'une autre) pour éviter de demander à l'utilisateur deux fois la même information.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet *Application* actif.
- *StoreSucceeded* fera référence à un objet *BooleenVariable* actif.
- La liste *InVariables* sera mise à une liste de références non vide dans le but d'activer des objets variables de n'importe quels types.

Description de la syntaxe:

`StorePersistent`	`-->`	`Target,`
		`StoreSucceeded,`
		`InVariables,`
		`OutFileName`
`Target`	`-->`	`GenericObjectReference`
`StoreSucceeded`	`-->`	`ObjectReference`
`InVariables`	`-->`	`ObjectReference+`
`OutFileName`	`-->`	`GenericOctetString`

ReadPersistent (*ReadSucceeded*, *OutVariables*, *InFileName*)

Lecture permanente (Lecture réussie, Variables de sortie, Nom de fichier d'entrée)

Demande au moteur MHEG-5 de lire la donnée ayant été sauvegardée par l'action StorePersistent.

La donnée à lire est récupérée à travers un ensemble d'objets *Variables*, référencés par le paramètre *OutVariables*.

La donnée est sauvegardée dans une structure de donnée de fichier. Le nom *InFileName* est un autre paramètre de l'action *ReadPersistent*. La présente Recommandation ne définit pas les nature, structure, propriété, protection ou restriction de l'espace de fichier. Cependant les actions *StorePersistent* et *ReadPersistent* utiliseront le même espace de nom de fichiers.

L'effet de l'action *ReadPersistent* est synchrone. Sur achèvement réussi de l'action *ReadPersistent*, la variable référencée par *ReadSucceeded* sera mise à *True*, sinon à *False*.

Exemple: considérons l'action élémentaire suivante comme partie de l'attribut *OnRestart* d'une Application:

:ReadPersistent (("myApp" 0) ("myApp" 1) (("scene1" 1) ("scene1" 2)) "myfile.txt")

Les contenus des variables indiquées par ("scene1" 1) et ("scene1" 2) seront mis aux valeurs lues dans le fichier "myfile.txt" au lancement de l'application.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet *Application* actif;
- *ReadSucceeded* fera référence à un objet *BooleenVariable* actif;
- la liste *OutVariables* sera mise à une liste de références non vide dans le but d'activer des objets variables de n'importe quels types.

Description de la syntaxe:

```
ReadPersistent       -->   Target,
                           ReadSucceeded,
                           OutVariables,
                           InFileName
Target               -->   GenericObjectReference
ReadSucceeded,       -->   ObjectReference
OutVariables         -->   ObjectReference+
InFileName           -->   GenericOctetString
```

Launch

(Lancement)

Active une nouvelle application en virant l'application actuellement active, s'il y en a une.

Exécute de manière synchrone la séquence d'actions suivantes:

1) appliquer le comportement *Destruction* de l'objet Scene actif en cours, s'il existe;
2) applique le comportement *Destruction* de l'objet *Application* actif en cours, s'il existe;
3) appliquer le comportement *Activation* de l'objet *Application* auquel l'action *Launch* a été adressée.

NOTE – Tous les événements générés pendant l'exécution de ces étapes sont mis en file d'attente et traités seulement à la fin de la séquence entière.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Application* non disponible.

Description de la syntaxe:

```
Launch               -->   Target
Target               -->   GenericObjectReference
```

Spawn

(Déposer temporairement)

Active une nouvelle application de sorte que l'application en cours soit démarrée à nouveau lors de la fin de la nouvelle application.

Exécute de manière synchrone la séquence d'actions suivantes:

1) exécuter l'action *OnSpawnCloseDown* de l'objet *Application* actif en cours;
2) stocker l'identificateur *GroupIdentifier* de l'application active en cours sur la pile d'identificateurs de l'application, si elle existe;
3) exécuter l'effet de l'action *Launch* conformément à la description précédente.

La pile d'identificateurs de l'application est un attribut optionnel du moteur MHEG-5. Si elle n'est pas implémentée, ou si elle est pleine, cette action sera implémentée comme une action *Launch*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Application* qui ne sera pas actif tant qu'il en existera déjà un.

Description de la syntaxe:

```
Spawn                 -->     Target
Target                -->     GenericObjectReference
```

Quit

(Quitter)

Ferme une application et redémarre l'application précédente.

Exécute de manière synchrone la séquence d'actions suivantes:

1) appliquer le comportement *Destruction* de l'objet *Scene* actif en cours, s'il existe;
2) appliquer le comportement *Destruction* de l'objet *Application* cible;
3) si le moteur MHEG-5 n'a pas implémenté de pile d'identificateur d'application, ou si cette pile est vide, le moteur MHEG-5 retournera alors à l'état *idle*. Dans tous les autres cas les étapes suivantes seront effectuées:
4) appliquer le comportement *Activation* à l'objet *Application* dont l'identificateur *GroupIdentifier* est sur le sommet de la pile d'identificateur de l'application. Il faut noter que ceci inclut l'exécution de l'action *OnStartUp* de cet objet;
5) oter le sommet de la pile d'identificateur de l'application;
6) exécuter l'action *OnRestart* de l'objet *Application* nouvellement activé.

Disposition pour l'utilisation:

- l'objet *Target* deviendra l'objet *Application* actif en cours.

Description de la syntaxe:

```
Quit                  -->     Target
Target                -->     GenericObjectReference
```

LockScreen

(Verrouillage écran)

Fige l'écran d'affichage et empêche la répercussion des changements sur les objets *Visibles*.

Exécute de manière synchrone la séquence d'actions suivantes:

1) incrémenter de 1 l'attribut interne *LockCount*;

2) si l'attribut *LockCount* a maintenant une valeur positive, verrouiller l'écran d'affichage.

Disposition pour l'utilisation:

- l'objet *Target* sera l'objet *Application* actif.

Description de la syntaxe:

```
LockScreen          -->     Target
Target              -->     GenericObjectReference
```

UnlockScreen

(Déverrouillage écran)

Cette action peut rafraîchir l'écran d'affichage et répercuter en une seule fois tous les changements sur les objets *Visibles*.

Exécute de manière synchrone la séquence d'actions suivantes:

1) décrémenter de 1 l'attribut interne *LockCount*. Si le résultat est négatif, mettre l'attribut *LockCount* à 0;

2) si l'attribut *LockCount* est égal à 0, rafraîchir l'écran d'affichage.

Disposition pour l'utilisation:

- l'objet *Target* sera l'objet Application actif.

Description de la syntaxe:

```
UnlockScreen        -->     Target
Target              -->     GenericObjectReference
```

OpenConnection (*OpenSucceeded*, *Protocol*, *Address*, *ConnectionTag*)

Tente d'ouvrir une connexion avec une entité située en dehors du moteur MHEG-5.

L'action *OpenConnection* possède les paramètres suivants:

Ouverture connexion (Ouverture réussie, Protocole, Adresse, Drapeau de connexion)

OpenSucceeded	Si l'action OpenConnection se termine avec succès, la variable référencée par *OpenSucceeded* sera mise à *True*, sinon à *False*.
Protocol	Identificateur du protocole à utiliser lors de l'établissement de la connexion.
Address	Adresse de la contrepartie avec laquelle la connexion devrait être établie. Le codage de ce paramètre dépend de la valeur de *Protocol*.
ConnectionTag	Entier utilisé pour référencer la connexion à l'intérieur de l'application.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet application actif.
- *OpenSucceeded* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
OpenConnection      -->     Target,
                            OpenSucceeded,
                            Protocol,
                            Address,
                            ConnectionTag
Target              -->     GenericObjectReference
OpenSucceeded       -->     ObjectReference
Protocol            -->     GenericOctetString
Address             -->     GenericOctetString□
ConnectionTag       -->     GenericInteger□
```

CloseConnection (*ConnectionTag*)

Tente de fermer une connexion avec une entité située en dehors du moteur MHEG-5.

L'action *CloseConnection* possède le paramètre suivant:

Fermeture de connexion (Balise de connexion)

ConnectionTag Entier référençant une connexion créée par l'action OpenConnection.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet application actif;
- si la connexion référencée par *ConnectionTag* n'est pas correctement établie, l'action *CloseConnection* est ignorée.

Description de la syntaxe:

```
CloseConnection     -->     Target,
                            ConnectionTag
Target              -->     GenericObjectReference
ConnectionTag       -->     GenericInteger
```

GetEngineSupport (*Feature, Answer*)

Obtient le soutien du moteur (attribut, réponse)

Renvoie un Booléen indiquant si le moteur MHEG-5 implémente l'option spécifique ou le jeu d'options de la présente Recommandation. Le résultat de cette action est renvoyé dans la variable *BooleanVariable* référencée par le paramètre *Answer* et peut être utilisé pour adapter le comportement de l'application au potentiel du moteur MHEG-5.

Feature est une chaîne codée selon l'ISO/CEI 646 et décrivant l'option ou le jeu d'options. Les chaînes autorisées sont définies ci-dessous; des chaînes supplémentaires peuvent être définies par le domaine applicatif. Ces chaînes sont *casesensitive* et des entiers vont remplacer les symboles N, W, H, X ou Y entre parenthèses.

La réponse à chacune de ces chaînes sera *True* ou *False*.

– AncillaryConnections

  demande si le moteur supporte les connexions point à point auxiliaires. Ces connexions concernent les actions *OpenConnection* et *CloseConnection*, et l'attribut *ConnectionTag* pour de nombreuses actions élémentaires.

– ApplicationStacking

  demande si le moteur supporte l'action *Spawn* de la classe *Application*

– Cloning

  demande si le moteur supporte l'action *Clone*

– FreeMovingCursor

  demande si le moteur supporte la classe *CursorShape*, les événements *CursorEnter* et *CursorLeave* ainsi que les actions *GetCursorPosition*, *SetCursorPosition* et *SetCursorShape*.

– MultipleAudioStreams(N)

  demande si le moteur supporte au moins N flux *Audio* simultanés

– MultipleRTGraphicsStreams(N)

  demande si le moteur supporte au moins N flux *RTGraphics* simultanés

– MultipleVideoStreams(N)

  demande si le moteur supporte au moins N flux *Video* simultanés

– OverlappingVisibles(N)

  demande si le moteur supporte au moins N objets *Visibles* recouvrants

– Scaling

  demande si le moteur MHEG-5 supporte les actions *ScaleBitmap* et *ScaleVideo*)

– SceneAspectRatio(W,H)

  demande si le moteur supporte un rapport d'aspect donné. W & H sont deux entiers, W/H étant le rapport d'aspect largeur sur hauteur

– SceneCoordinateSystem(X,Y)

  demande si le moteur supporte un système de coordonnées donné. X & Y sont deux entiers définissant le système de coordonnées

– TrickModes

  demande si le moteur supporte les modes *trick* pour les objets *Streams*

Dispositions pour l'utilisation:

- l'action *GetEngineSupport* sera adressée seulement à l'objet *Application* actif;
- *Answer* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
GetEngineSupport     -->     Target,
                     -->     Feature,
                             Answer
Target               -->     GenericObjectReference
Feature              -->     GenericOctetString
Answer               -->     ObjectReference
```

10.5 Description formelle

```
Application Class       -->   Group Class,
                              OnSpawnCloseDown?,
                              OnRestart?
                              DefaultAttributes?
OnSpawnCloseDown        -->   Action Class
OnRestart               -->   Action Class
DefaultAttributes       -->   DefaultAttribute+
DefaultAttribute        -->   CharacterSet | BackgroundColour |
                              TextColour | Font | FontAttributes |
                              BitmapContentHook |
                              InterchangedProgramContentHook |
                              StreamContentHook |
                              TextContentHook | LineArtContentHook |
                              ButtonRefColour | HighlightRefColour |
                              SliderRefColour
CharacterSet            -->   INTEGER
BackgroundColour        -->   Colour
TextColour              -->   Colour
Font                    -->   OctetString | ObjectReference
FontAttributes          -->   OctetString
BitmapContentHook       -->   INTEGER
StreamContentHook       -->   INTEGER
TextContentHook         -->   INTEGER
LineArtContentHook      -->   INTEGER
InterchangedProgram-    -->   INTEGER
ContentHook
ButtonRefColour         -->   Colour
HighlightRefColour      -->   Colour
SliderRefColour         -->   Colour
```

11 Classe Scene *(Scène)*

Description:	définit un ensemble d'objets *Ingredients* destinés à être activés ensemble
classe de base:	*Group*
Sous-classes:	sans objet
Etat:	classe concrète

11.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

11.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec une sémantique identique.

11.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

InputEventRegister

(Registre d'événement d'entrée)

Registre des événements *UserInput* autorisés pour cette *Scene*.

Tant que cette *Scene* est active, le moteur MHEG-5 générera les seuls événements *UserInput* ayant une donnée associée avec le contenu de ce registre *InputEventRegister*.

Les contenus des registres *InputEventRegisters* ne sont pas définis dans la présente Recommandation.

- Entier identifiant InputEventRegister.

Exemple: on pourrait définir deux *InputEventRegisters*: l'un dédié au déplacement du pointeur des événements d'entrée (*MouseClick*, etc.) et un autre dédié au contrôle distant des événements d'entrée (*Up, Down, Left, Right, Enter, Quit*, etc.). Le fait de savoir quels types d'événements d'entrée sont attendus par la scène permettra au moteur MHEG-5 d'utiliser les dispositifs physiques de l'utilisateur fournis à l'utilisateur pour générer de tels événements.

SceneCoordinate-System

(Système de coordonnée de scène)

Taille du système de coordonnées de cet objet *Scene*.

Cet attribut est exprimé en nombre de rangées et de colonnes.

- Deux entiers exprimant les tailles respectives de l'objet *Scene* en x et en y.

AspectRatio

(Format de scène)

Format de présentation d'origine de l'objet *Scene*. Cet attribut est exprimé en rapport de largeur sur hauteur.

- Fraction optionnelle.
- Valeur par défaut: 4/3.

MovingCursor

(Curseur de déplacement)

Indique si l'objet *Scene* attend un curseur à déplacement libre.

La prise en compte de ce type de curseur dans un moteur MHEG-5 est optionnelle. Cependant, un domaine applicatif MHEG-5 peut déclarer un telle prise en compte comme obligatoire. Un moteur ne la supportant pas laissera tomber cet attribut et agira de la manière suivante:

- lorsque cet attribut est à *False*, le moteur n'affichera pas de curseur sur l'écran;
- lorsque cet attribut est à *True*, le moteur affichera un curseur sur l'écran. L'utilisateur pourra déplacer ce curseur dans toutes les positions à l'intérieur de l'espace de coordonnées de l'objet *Scene*. Lorsque le curseur rentre dans les (ou sort des) limites de la boîte d'un objet *Interactible*, un événement *CursorEnter* (*CursorLeave*) sera généré pour cet *Interactible*. Le moteur supportera les actions *SetCursorPosition* et *GetCursorPosition*;

Ce qui suit s'applique à tout attribut en général:

- Booléen optionnel;
- valeur par défaut: *False*.

NextScenes

(Scènes suivantes)

Une liste optionnelle d'*OctetStrings*, qui sera interprétée comme des identificateurs *GroupIdentifiers* d'objets *Scene* pouvant être présentés après celui-ci, selon un facteur de pondération mesurant la vraisemblance pour ces objets *Scenes* d'être effectivement présentés. Le facteur de pondération sera un entier compris dans l'intervalle [0, 255], 255 indiquant la plus forte vraisemblance. Ceci peut être utilisé par le moteur MHEG-5 pour résoudre les conflits de cache ou de téléchargement anticipé.

11.1.3 Attributs internes propres

Cette classe définit l'attribut interne complémentaire suivant:

Timers

(Chronomètres)

Liste des *Timers* représentant les positions temporelles où l'objet *Scene* recevra les événements *TimerFired*.

Chaque *Timer* possède un numéro d'identité unique à l'intérieur de la liste des *Timers*. Une position temporelle est exprimée en millièmes de seconde et mesurée à partir de l'origine de temps du *Timer*. Un *Timer* sera créé en exécutant l'action *SetTimer* sur l'objet *Scene* actif. L'origine de temps du *Timer* est par défaut la position dans le temps correspondant à l'exécution de l'action *SetTimer*. Cependant, si le booléen *AbsoluteTime* est codé et vaut *True*, l'origine de temps du *Timer* est la position dans le temps correspondant à la génération de l'événement *IsRunning* de l'objet *Scene*.

- Séquence des structures de données suivantes:
  - identificateur de *Timer*: entier;
  - position de *Timer*: entier;
  - *AbsoluteTime*: Booléen optionnel, par défaut à *False*.
- Valeur initiale: séquence vide.

11.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, les événements suivants sont définis:

UserInput

(Saisie utilisateur)

Cet événement sera généré par le moteur MHEG-5 pour indiquer qu'une saisie de l'utilisateur a été effectuée.

- Donnée associée: *UserInputEventTag* – Entier. La valeur de la donnée associée sera en cohérence avec le contenu de l'attribut *InputEventRegister*.

TimerFired

(Amorçage chronomètre)

Cet événement est généré lors du lancement d'un *Timer*.

- Donnée associée: *TimerIdentifier* – Entier.

11.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base avec une sémantique identique.

11.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

TransitionTo (*ConnectionTag*, *TransitionEffect*)

Vérifie que l'objet *Scene* adressé est différent de l'objet *Scene* actif. S'il ne l'est pas, on ignore l'action; s'il l'est, ôte l'objet *Scene* actif de l'écran et le remplace par l'objet *Scene* adressé comme suit:

Transition vers (Drapeau de connexion, Effet de transition)

Exécute de manière synchrone les séquences d'actions suivantes:

1) appliquer le comportement *Deactivation* à tous les objets *Ingredients* actifs de l'objet *Application* courant ayant le paramètre *Shared* mis à *False* dans l'ordre inverse de celui-ci de l'attribut *Items* de l'objet Application;
2) appliquer les comportements *Deactivation* et *Destruction* à l'objet *Scene* actif en cours, s'il y en a un. (Ceci démarre le *TransitionEffect*);
3) appliquer le comportement *Preparation* à l'objet *Scene* auquel l'action *TransitionTo* a été adressée;
4) appliquer le comportement *Activation* à l'objet *Scene* auquel l'action *TransitionTo* a été adressée. (Ceci arrête le *TransitionEffect.*) La base de temps du nouvel objet *Scene* démarre après la génération de l'événement *IsRunning*.

Cette action a un paramètre optionnel appelé *ConnectionTag.* Si ce paramètre n'est pas codé, la référence à l'objet *Scene* sera résolue à l'intérieur de l'espace de nom de l'objet *Application* actif.

Si le paramètre *ConnectionTag* est codé, la référence à l'objet *Scene* cible et toutes les références *ContentReference* faites à partir de l'objet *Scene* seront résolues à l'intérieur d'un espace de nom utilisé pour la communication à travers un lien de communication avec la balise *ConnectionTag* (voir l'action *OpenConnection* dans la classe *Application*).

De plus cette action possède un paramètre *TransitionEffect*, qui détermine le type d'effet visuel de transition à implémenter lorsque l'action *TransitionTo* est effectuée. Le fait d'implémenter un effet de transition quelconque est optionnel pour le moteur MHEG-5. Le codage de l'attribut *TransitionEffect* doit être spécifié par le domaine applicatif.

NOTE – Dans le cas de certaines transitions, on peut demander au moteur de piloter le comportement *Destruction* des objets *Visibles* différemment d'autres *Deactivation* (n'étant pas introduit par *TransitionTo*). Par exemple, la représentation visuelle de l'objet *Scene* en cours peut être sauvegardée dans le sous-système graphique pour un effet *Push* ou *Wipe* dans lequel la première scène est progressivement remplacée par la seconde.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Scene* non disponible.

Description de la syntaxe:

```
TransitionTo          -->    Target,
                             ConnectionTag?,
                             TransitionEffect?
Target                -->    GenericObjectReference
ConnectionTag         -->    GenericInteger
TransitionEffect      -->    GenericInteger
```

SetTimer (*TimerId*, *TimerValue*, *AbsoluteTime*)

Affectation de Chronomètre (Identificateur de chrono, Valeur de chrono, Temps absolu)

Met à jour la liste des chronomètres de l'objet *Scene*.

Exécute la séquence d'actions suivante:

1) mettre à jour l'attribut interne *Timers* de la *Scene* selon les règles suivantes:
   a) si *TimerId* est l'identificateur d'un *Timer* existant de l'objet *Scene*, remplacer le précédent. Le paramètre *AbsoluteTime* est ignoré; en d'autres mots, un *Timer* absolu ne peut pas être remplacé par un *Timer* relatif à l'objet *Scene*;
   b) s'il n'existe pas de *Timer* ayant ce *TimerId* dans l'objet *Scene*, insérer un nouveau *Timer* avec l'identificateur *TimerId* et les valeurs *TimerValue* et *AbsoluteTime* dans l'objet *Scene*. Si *AbsoluteTime* n'est pas codé, il est mis à *False* par défaut;
   c) si *TimerValue* n'est pas codé et s'il y a un *Timer* avec l'identificateur *TimerId* dans l'objet *Scene*, ôter ce *Timer* de la liste *Timers*;

d) si *TimerValue* n'est pas codé et s'il n'y a pas de *Timer* avec l'identificateur *TimerId* dans la liste *Timers*, ignorer cette action.

2) L'objet *Scene* actif recevra les événements *TimerFired* selon la nouvelle valeur de la liste *Timers*.

Si *AbsoluteTime* vaut *True*, le paramètre *TimerValue* de cette action sera interprété comme un déplacement de temps à partir du moment de début d'activité de l'objet *Scene*. Sinon, le paramètre *TimerValue* de cette action sera interprété comme un déplacement de temps depuis le moment d'invocation de l'action. Dans les deux cas, la mesure sera faite en millièmes de seconde.

Si le paramètre *TimerValue* vaut zéro et *AbsoluteTime* vaut *False*, le *Timer* sera lancé immédiatement.

Le fait d'enlever ou de changer un *Timer* ne supprime pas les événements en instance du *Timer* précédent.

Disposition pour l'utilisation:

- l'objet *Target* sera l'objet *Scene* actif.

Description de la syntaxe:

```
SetTimer            -->    Target,
                           TimerId,
                           TimerValue?,
                           AbsoluteTime?
Target              -->    GenericObjectReference
TimerId             -->    GenericInteger
TimerValue          -->    GenericInteger□
AbsoluteTime        -->    GenericBoolean
```

SendEvent (*EmulatedEvent-Source*, *EmulatedEventType*, *EmulatedEventData*)

Emission d'événement (Source événement émulé, Type d'événement émulé, Donnée d'événement émulé)

Force l'apparition d'un événement

Exécute la séquence d'actions suivante:

1) générer un événement correspondant à *EmulatedEventType*, *EmulatedEventSource* et *EmulatedEventData* comme s'il avait été généré tout à fait normalement;

2) stocker cet événement dans la file d'attente d'événements synchrones ou asynchrones, selon le type *EmulatedEventType*.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet *Scene* actif;
- *EmulatedEventSource* fera référence à un objet MHEG-5 compatible avec *EmulatedEventType*;
- *EmulatedEventData* sera soit une valeur directe, ou une référence à un objet *Variable* actif d'un type compatible avec la donnée associée de *EmulatedEventType*.

Description de la syntaxe:

```
SendEvent             -->    Target,
                             EmulatedEventSource,
                             EmulatedEventType,
                             EmulatedEventData?
Target                -->    GenericObjectReference
EmulatedEventSource   -->    GenericObjectReference
EmulatedEventType     -->    IsAvailable | ContentAvailable |
                             IsDeleted | IsRunning | IsStopped
                             | UserInput | AnchorFired |
                             TimerFired | AsynchStopped |
                             InteractionCompleted | TestEvent
                             | TokenMovedFrom | TokenMovedTo |
                             FirstItemPresented |
                             LastItemPresented | HeadItems |
                             TailItems | ItemSelected |
                             ItemDeselected | StreamEvent |
                             StreamPlaying | StreamStopped |
                             CounterTrigger | HighlightOn |
                             HighlightOff | CursorEnter |
                             CursorLeave | IsSelected |
                             IsDeselected | EntryFieldFull
EmulatedEventData     -->    GenericBoolean | GenericInteger |
                             GenericOctetString□
```

SetCursorShape (*NewCursorShape*)

Affecte une forme de curseur (Nouvelle forme de curseur)

Affecte la forme du curseur "libre déplacement".

Cette action aura un effet seulement si l'option de curseur "libre déplacement" est implémentée par le moteur MHEG-5.

Si le paramètre *NewCursorShape* n'est pas codé, le curseur est ôté de l'objet *Scene*.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet *Scene* actif;
- *NewCursorShape* fera référence à un objet *CursorShape* actif.

Description de la syntaxe:

```
SetCursorShape        -->    Target,
                             NewCursorShape?
Target                -->    GenericObjectReference
NewCursorShape        -->    GenericObjectReference
```

SetCursorPosition (*XCursor, YCursor*)

Affecte la position de curseur (X-curseur, Y-Curseur)

Affecte la position du curseur "libre déplacement".

Cette action aura un effet seulement si l'option de curseur "libre déplacement" est implémentée par le moteur MHEG-5.

Exécute la séquence d'actions suivante:

1) mettre la position du pointeur de curseur à l'intérieur de l'espace de coordonnées de l'objet *Scene*;

2) générer les événements *CursorLeave* et *CursorEnter* si les objets *Interactible* sont affectés par l'effet de cette action.

Si un Interactible B recouvre un autre A, un *SetCursorPosition* d'un point dans A (non dans B) vers un point à l'intérieur de la zone de recouvrement générera *CursorLeave(A)* et un *CursorEnter(B).*

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet *Scene* actif;
- *Xcursor* et *YCursor* correspondent à un emplacement à l'intérieur du rectangle défini par l'attribut *SceneCoordinateSystem* de la *Scene* active.

Description de la syntaxe:

```
SetCursorPosition    -->    Target,
                            XCursor,
                            YCursor
Target               -->    GenericObjectReference
XCursor              -->    GenericInteger
YCursor              -->    GenericInteger
```

GetCursorPosition (*XOut, YOut*)

Récupère la position de curseur (X-sortie, Y-sortie)

Affecte la position du curseur dans l'espace aux objets *Variables* référencées par *XOut* et *YOut* de coordonnées de l'objet *Scene.*

Cette action aura un effet seulement si l'option de curseur "libre déplacement" est implémentée par le moteur MHEG-5.

Dispositions pour l'utilisation:

- l'objet *Target* sera l'objet *Scene* actif;
- *XOut* et *YOut* feront référence à des objets *IntegerVariable*.

Description de la syntaxe:

```
GetCursorPosition    -->   Target,
                           XOut,
                           Yout
Target               -->   GenericObjectReference
Xout                 -->   ObjectReference
Yout                 -->   ObjectReference
```

11.5 Description formelle

```
Scene Class             -->  Group Class,
                             InputEventRegister,
                             SceneCoordinateSystem,
                             AspectRatio?,
                             MovingCursor?,
                             NextScenes?
InputEventRegister      -->  INTEGER
SceneCoordinateSystem   -->  XScene, Yscene
Xscene,                 -->  INTEGER
Yscene                  -->  INTEGER
AspectRatio             -->  Width, Height□
Width                   -->  INTEGER□
Height                  -->  INTEGER□
MovingCursor            -->  BOOLEAN□
NextScenes              -->  NextScene+
NextScene               -->  SceneRef,
                             SceneWeight
SceneRef                -->  OctetString
SceneWeight             -->  INTEGER
```

12 Classe *Ingredient* (Ingrédient)

Description: définit les fonctionnalités associées aux classes permettant de décorer les *Scenes* et les *Applications*

classe de base: *Root*

Sous-classes: *Link*, *Program*, *Palette*, *Font*, *CursorShape*, *Variable*, *Presentable*

Etat: classe abstraite

12.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

12.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ObjectIdentifier	*Root*	La partie ObjectNumber de cet attribut sera unique à l'intérieur du groupe d'appartenance de cet objet et ne sera pas égale à 0. Si l'attribut GroupIdentifier est codé, il lui sera affecté le GroupIdentifier du groupe d'appartenance de cet objet.

12.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

InitiallyActive
(Actif à l'origine)

Ce paramètre est utilisé pour déterminer quels sont les objets actifs à l'origine dans une *Scene* ou une *Application*.

- Booléen optionnel.
- Valeur par défaut: *True*.

ContentHook
(Crochet de contenu)

Détermine le format de codage de la donnée incluse ou référencée par l'attribut *Content*.

- Entier optionnel.
- Valeur par défaut: dépend des sous-classes, la valeur codée par *Application* dans l'un des attributs suivants: *BitmapContentHook*, *LineArtContentHook*, *InterchangedProgramContentHook*, *StreamContentHook* et *TextContentHook*.

OriginalContent
(Contenu d'origine)

Valeur de l'attribut *Content* au moment de la préparation.

Cet attribut contient soit une donnée incluse ou une référence à une source de donnée externe.

- *OriginalContent* est un Attribut optionnel.
- Type de donnée: *ReferencedContent* ou *IncludedContent*.
- Valeur par défaut: sans objet.

La donnée incluse est codée directement dans un *OctetString*.

Une référence à une source de donnée externe est composée de:

1) un *ContentReference*, formé d'un *OctetString* référençant une donnée externe.
   - Type de donnée: *OctetString*;

2) un *ContentSize* optionnel, représentant la taille en octets de la source externe. Cet attribut pourrait être utilisé par le moteur MHEG-5 pour déterminer le nombre de ressources demandées pour restituer l'objet *Ingredient*. Il est laissé à la responsabilité de l'application de s'assurer de la compatibilité de l'attribut *ContentSize* avec la taille réelle de la source externe de donnée.
   - *ContentSize* est un Attribut optionnel.
   - Type de donnée: *Entier*.
   - Valeur par défaut: aucune.

   Exemple: on considère un objet *Bitmap* dont l'attribut *OriginalContent* contient une référence à un fichier de données. Son attribut *ContentSize* devrait contenir une estimation de la taille des données externes;

3) un *ContentCachePriority* optionnel, représentant le degré de pertinence pour cacher cette source de données externes. Cet attribut peut être comparé par le moteur MHEG-5 à d'autres attributs *ContentCachePriority* pour déterminer quelles sont les sources de données externes ayant la plus forte probabilité d'être demandées à nouveau par l'application. Une valeur plus haute indique un degré de probabilité plus élevé. La gestion de ces priorités selon un intervalle cohérent est laissée à l'appréciation de l'application. Il est recommandé au moteur MHEG-5 de cacher de préférence les sources de données externes ayant une haute priorité.

- *ContentCachePriority* est un Attribut optionnel.
- Type de donnée: *Entier* compris dans l'intervalle [0, 255].
- Valeur par défaut: 127.
- Valeur spécifique: 0 signifie que le cache n'est pas autorisé pour les contenus externes référencés par cet objet *Ingredient*.

Shared

(Partagé)

Indique si l'objet *Ingredient* continue à être présenté tout au long d'une transition d'objet *Scene*. Il est utilisé pour empêcher la destruction d'objets utilisés dans des scènes consécutives. Plus spécialement, lorsqu'une action *TransitionTo* est adressée à une scène *B* tandis qu'une scène *A* est active, tous les objets *Ingredients* de l'objet *Application* actif sont automatiquement désactivés sauf ceux dont l'attribut *Shared* vaut *True*.

Disposition pour l'utilisation:

- si l'objet *Ingredient* est un élément d'un objet *Scene* ou *Template*, l'attribut *Shared* ne sera pas codé.

Synopsis:

- Booléen optionnel;
- valeur par défaut: *False*.

12.1.3 Attributs internes propres

Cette classe définit les attributs internes complémentaires suivants:

Content

(Contenu)

Cet attribut contient soit une donnée incluse ou une référence à une source de donnée externe.

La donnée incluse est codée directement dans un *OctetString*.

Une référence à une source de donnée externe est composée de:

1) un *ContentReference*, formé d'un *OctetString* référençant une donnée externe.

- Type de donnée: *OctetString*.
- Valeur initiale: *ContentSize* de l'attribut *OriginalContent*;

2) un *ContentSize* optionnel, représentant la taille en octets de la source externe.

- *ContentSize* est un Attribut optionnel.
- Type de donnée: *Entier*.
- Valeur initiale: *ContentSize* de l'attribut *OriginalContent*;

3) un attribut *ContentCachePriority* optionnel, représentant le degré de pertinence pour cacher cette source de donnée externe.

- *ContentCachePriority* est un Attribut optionnel.
- Type de donnée: *Entier* compris dans l'intervalle [0, 255].
- Valeur initiale: *ContentCachePriority* de l'attribut *OriginalContent*.
- Valeur spécifique: 0 signifie que le cache n'est pas autorisé pour les données *content* externes référencées par cet objet *Ingredient*.

L'attribut *Content* ne sera pas défini pour les classes spécifiant l'absence de codage de l'attribut *OriginalContent*.

- Attribut optionnel.
- Type de donnée: *ReferencedContent* ou *IncludedContent*.
- Valeur initiale: valeur de *OriginalContent*.

12.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

12.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de l'objet:

Destruction (Destruction)

Exécute la séquence d'actions suivante:

1) si l'attribut *Content* est mis à une référence de source de donnée externe et si *ContentCachePriority* vaut 0, le moteur MHEG-5 libérera toutes les ressources allouées à la donnée *Content* externe de cet objet *Ingredient*;

2) si l'attribut *Content* est mis à une référence de source de donnée externe et si *ContentCachePriority* est différent de 0, le moteur MHEG-5 peut libérer toutes les ressources allouées à la donnée *Content* externe de cet objet *Ingredient* ou les mettre dans le cache;

3) appliquer le comportement *Destruction* comme hérité de la classe *Root*.

12.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData (*NewContent*)

Affecte à l'attribut *Content* de l'objet *Ingredient* destination le contenu *NewContent*.

Affecter donnée (Nouveau contenu)

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Ingredient* disponible;
- l'attribut *ContentHook* de l'objet *Ingredient* destination sera codé;
- la donnée incluse ou référencée dans *NewContent* aura un format de codage déterminé par l'attribut *ContentHook* de l'objet *Ingredient* destination;
- si le contenu *Content* contient actuellement une donnée incluse, *NewContent* sera affecté ou référencera une donnée incluse;
- si le contenu *Content* contient actuellement une référence à une source de donnée externe, *NewContent* sera affecté ou référencera une référence à une source de donnée externe.

Description de la syntaxe:

```
SetData                  -->  Target,
                              NewContent
Target                   -->  GenericObjectReference
NewContent               -->  NewIncludedContent |
                              NewReferencedContent
NewIncludedContent       -->  GenericOctetString
NewReferencedContent     -->  NewContentReference,
                              NewContentSize?,
                              NewContentCachePriority?
NewContentReference      -->  GenericContentReference
NewContentSize           -->  GenericInteger
NewContentCachePriority  -->  GenericInteger
```

Clone (*CloneRefVar*)

L'effet de cette action est décrit ci-dessous pour les moteurs supportant l'option *Cloning*. Les moteurs ne la supportant pas devront l'ignorer.

Clone (Variable référence à clone)

Cette action copie la destination *Target* en ajoutant la copie au groupe contenant déjà la destination *Target* et en utilisant une référence *ObjectReference* unique obtenue du moteur. La référence *ObjectReference* faisant référence à la copie est renvoyée dans la référence *ObjectRefVariable* référencée par *CloneRefVar*.

Exécute la séquence d'actions suivante:

1) déterminer une référence ObjectReference à l'intérieur du groupe même de *Target*;

2) créer une copie de *Target*, prenant en compte les seuls attributs échangés et non les attributs internes. L'attribut *ObjectIdentifier* hérité de la classe *Root* n'est pas copié, mais affecté à la référence *ObjectReference* déterminée à l'étape 1;

3) ajouter la copie au groupe contenant l'objet *Target*;

4) affecter à la référence *ObjectRefVariable* référencée par *CloneRefVar* la référence *ObjectReference* déterminée à l'étape 1;

5) Applique le comportement *Preparation* à la copie.

Un objet créé au moyen de cette action est activé/désactivé lorsque le groupe contenant cet objet est activé/détruit. Les objets sont désactivés/détruits avec d'autres objets *Ingredients* statiques dans leur ordre inverse de création. Un objet *Ingredient* dynamiquement créé peut aussi être détruit au moyen de l'action élémentaire *Unload* si son attribut *Content* est différent de *Null*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Ingredient* disponible;
- *CloneRefVar* sera une référence *ObjectRefVariable* active.

Description de la syntaxe:

```
Clone                 -->    Target,
                             CloneRefVar
Target                -->    GenericObjectReference
CloneRefVar           -->    ObjectReference
```

Preload

(Chargement anticipé)

Prépare un objet *Ingredient* et fournit au moteur MHEG-5 l'occasion de préparer la donnée *Content* d'un objet *Ingredient* en vue de son usage ultérieur.

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Preparation*;

2) le moteur MHEG-5 peut de manière optionnelle récupérer et/ou décoder la donnée *Content* associée à l'objet *Ingredient* destination.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Ingredient* non disponible;
- l'attribut *Content* de l'objet *Ingredient* destination sera différent de *Null*.

Description de la syntaxe:

```
Preload               -->    Target
Target                -->    GenericObjectReference
```

Unload

(Déchargement)

Détruit un objet *Ingredient* et fournit au moteur MHEG-5 l'occasion de libérer des ressources allouées à un *Ingredient*.

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Destruction*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Ingredient* inactif et disponible;
- l'attribut *Content* de l'objet *Ingredient* destination sera différent de *Null*.

Description de la syntaxe:

```
Unload                  -->     Target
Target                  -->     GenericObjectReference
```

12.5 Description formelle

```
Ingredient Class            --> Root Class,
                                InitiallyActive?,
                                ContentHook?,
                                OriginalContent?,
                                Shared?
InitiallyActive             --> BOOLEAN□
ContentHook                 --> INTEGER
OriginalContent             --> IncludedContent |
                                ReferencedContent
IncludedContent             --> □□□□□□□□□□□□
ReferencedContent           --> ContentReference,
                                ContentSize?,
                                ContentCachePriority?
ContentSize                 --> INTEGER□
ContentCachePriority        --> INTEGER□
Shared                      --> BOOLEAN
```

13 Classe *Link* (Lien)

Description: définit les fonctionnalités associées à la réaction aux événements qui conduisent à la réalisation d'une séquence d'actions élémentaires

classe de base: *Ingredient*

Sous-classes: aucune

Etat: classe concrète

13.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

13.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.

13.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

LinkCondition

(Condition de lien)

La condition de lien *LinkCondition* est composée d'un *EventSource*, un *EventType* et un *EventData*.

Lorsqu'un événement émanera d'un objet, le moteur MHEG-5 lancera un lien spécifique si et seulement si:

- l'objet *Link* est actif;
- *EventSource* est égal à la référence à objet de l'objet d'où l'événement émane (L'identificateur *GroupIdentifier* appartient par défaut au *GroupIdentifier* de l'objet *Group* dans lequel le *Link* lui-même est inclus;
- *EventType* est égal au type de l'événement apparu;
- *soit, EventData* est égal à la valeur de donnée fournie avec l'événement *ou EventData* n'est pas codé.

NOTE – Le type de donnée passé avec chaque événement est décrit au paragraphe 53. Le lancement d'un objet *Link* conduit à l'exécution de son *LinkEffect*.

LinkEffect

(Effet de lien)

Insertion d'un objet *Action*.

Lorsque le *Link* est lancé, les actions élémentaires comprises à l'intérieur de cet objet *Action* sont exécutées de manière synchrone.

13.1.3 Attributs internes propres

Cette classe ne définit aucun attribut interne supplémentaire.

13.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

13.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet.

Activation (Activation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Activation* comme hérité de la classe de base;
2) rendre l'objet *Link* réceptif aux événements satisfaisant à sa *LinkCondition*;
3) mettre l'état *RunningStatus* de l'objet *Link* à *True*;
4) générer l'événement *IsRunning*.

Deactivation (Désactivation)

Exécute la séquence d'actions suivante:

1) mettre l'objet *Link* dans l'état inactif pour le rendre non réceptif aux événements;
2) appliquer le comportement *Deactivation* comme hérité de la classe de base.

13.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

Activate

(Active)

Rend un *Link* réceptif aux événements correspondants à sa condition *LinkCondition*.

Exécute les actions suivantes:

1) si l'objet *Link* destination est actif, ne pas tenir compte de cette action;
2) si l'objet *Link* destination est inactif, appliquer le comportement *Activation* de l'objet *Link*.

Description de la syntaxe:

```
Activate            -->     Target
Target              -->     GenericObjectReference
```

Deactivate

(Désactive)

Rend un objet *Link* non réceptif aux événements.

Exécute les actions suivantes:

1) si l'objet *Link* destination est inactif, ne pas tenir compte de cette action;
2) si l'objet *Link* destination est actif, appliquer le comportement *DeActivation* de l'objet *Link*.

Disposition pour l'utilisation:

- l'objet *Link* destination sera un objet *Link* disponible.

Description de la syntaxe:

```
Deactivate          -->     Target
Target              -->     GenericObjectReference
```

13.5 Description formelle

```
Link Class              -->  Ingredient Class,
                             LinkCondition,
                             LinkEffect
LinkCondition           -->  EventSource,
                             EventType,
                             EventData?
LinkEffect              -->  Action Class
EventSource             -->  ObjectReference
EventType               -->  IsAvailable | ContentAvailable |
                             IsDeleted | IsRunning | IsStopped
                             | UserInput | AnchorFired |
                             TimerFired | AsynchStopped |
                             InteractionCompleted | TestEvent
                             | TokenMovedFrom | TokenMovedTo |
                             FirstItemPresented |
                             LastItemPresented | HeadItems |
                             TailItems | ItemSelected |
                             ItemDeselected | StreamEvent |
                             StreamPlaying | StreamStopped |
                             CounterTrigger | HighlightOn |
                             HighlightOff | CursorEnter |
                             CursorLeave | IsSelected |
                             IsDeselected □□EntryFieldFull |
                             EngineEvent
EventData               -->  OctetString | BOOLEAN | INTEGER
```

14 Classe *Program* (Programme)

Description: définit la manière de gérer l'exécution de code procédural externe

classe de base: *Ingredient*

Sous-classes: *RemoteProgram, ResidentProgram, InterchangedProgram*

Etat: classe abstraite

14.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

14.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
InitiallyActive	*Ingredient*	Cet attribut est obligatoire pour cette classe et sera mis à False.

14.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Name

(Nom)

Nom du code procédural externe à appeler lors de l'activation de l'objet *Program*.

Le mappage entre l'attribut *Name* et le nom réel du code procédural externe n'est pas définidans la présente Recommandation.

- *OctetString*.

InitiallyAvailable

(Disponible initialement)

Ce paramètre est utilisé pour déterminer les objets *Program* dans un objet *Scene* ou *Application* devant être initialement préparés.

- Booléen optionnel;
- valeur par défaut: *True*.

14.1.3 Attributs internes propres

Cette classe ne définit aucun attribut interne supplémentaire.

14.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, l'événement suivant est défini:

AsynchStopped

(Fin asynchrone)

Cet événement sera généré lorsqu'un objet *Program* exécuté via l'action *Fork* en a terminé avec son exécution.

- Pas de donnée associée.

14.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet:

Activation

(Activation)

1) si cela n'a pas été fait durant la préparation, rechercher l'emplacement du code procédural externe au moyen de l'attribut *Name*;
2) si le code procédural externe n'est pas trouvé, ne pas tenir compte de cette action. Sinon:
3) affecter les paramètres du code procédural externe de l'objet *Program* comme indiqué dans *Parameters*;
4) appliquer le comportement *Activation* comme hérité de la classe de base;
5) lancer l'exécution de manière synchrone ou asynchrone du code procédural externe selon l'action invoquant l'exécution;
6) mettre l'attribut *RunningStatus* à *True*;
7) générer l'événement *IsRunning*.

Deactivation

(Désactivation)

Si l'attribut *RunningStatus* est à *False*, ignorer cette action. Sinon, exécuter la séquence d'actions suivante:

1) forcer la fin d'exécution du programme;

2) appliquer le comportement *Deactivation* comme hérité de sa classe de base.

NOTE – Le comportement *Deactivation* génère l'événement *IsStopped* comme définit dans *Root*.

14.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData

(Affectation de donnée)

Cette action ne sera jamais appliquée à un objet d'une sous-classe de la classe *Program*.

Call (*CallSucceeded*, *Parameters*)

Appel (Appel réussi, Paramètres)

Demande l'exécution d'un code procédural externe et attend sa fin d'exécution.

Exécute la séquence d'actions suivante:

1) si l'objet *Program* est non disponible, appliquer le comportement *Preparation*;

2) si l'objet *Program* est actif, ne pas tenir compte de cette action. Sinon:

3) appliquer le comportement *Activation*;

4) attendre la fin de l'exécution du code procédural externe. Si le *Program* se termine anormalement mettre la variable référencée par *CallSucceeded* à *False*; sinon à *True*;

5) mettre les valeurs des variables référencées par *Parameters* aux valeurs renvoyées par *Program* (celles-ci pourront être non valides si *CallSucceeded* est à *False*);

6) appliquer le comportement *Deactivation*.

Dispositions pour l'utilisation:

- *CallSucceeded* sera affecté à un objet *BooleanVariable* actif;
- les paramètres *Parameters* seront affectés à une liste de valeurs correspondant aux paramètres attendus d'un code procédural externe. L'ordre de la liste *Parameters* correspondra à l'ordre des paramètres du code procédural externe. Les paramètres passés par valeur seront affectés directement à la valeur correspondante. Les paramètres passés par référence seront passés via des objets *Variable* actifs d'un type de donnée adéquat.

Description de la syntaxe:

```
Call              -->        Target,
                             CallSucceeded,
                             Parameters?
```

```
Target            -->        GenericObjectReference

CallSucceeded     -->        ObjectReference

Parameters        -->        Parameter+

Parameter         -->        GenericBoolean | GenericInteger |
                             GenericOctetString |
                             GenericObjectReference |
                             GenericContentReference
```

Fork (*ForkSucceeded*, *Parameters*)

Bifurcation (Succès, paramètres)

Demande l'exécution d'un code procédural externe sans en attendre la fin d'exécution.

Exécute la séquence d'actions suivante:

1) si l'objet *Program* n'est pas disponible, appliquer le comportement *Preparation*;
2) si l'objet *Program* est actif, ne pas tenir compte de cette action. Sinon:
3) applique le comportement *Activation*;
4) repasser le contrôle au moteur MHEG-5 sans attendre la fin de l'exécution du code procédural externe.

Lorsque l'exécution du code procédural externe est terminée, exécute la séquence d'actions suivante:

1) si le programme se termine anormalement, mettre la variable référencée par *ForkSucceeded* à *False*; sinon la mettre à *True*;
2) mettre les valeurs des variables référencées par *Parameters* aux valeurs renvoyées par *Program* (celles-ci peuvent être non valides si *ForkSucceeded* vaut *False*);
3) appliquer le comportement *Deactivation*;
4) générer l'événement *AsynchStopped*.

NOTE – Les paramètres peuvent être modifiés par l'objet *Program*; dans ce cas, ces paramètres resteront indéfinis tant que l'objet *Program* ne s'est pas normalement terminé, c'est-à-dire jusqu'à la génération de *AsynchStopped*.

Dispositions pour l'utilisation:

- *ForkSucceeded* sera mis à un objet *BooleanVariable* actif;
- les paramètres *Parameters* seront affectés à une liste de valeurs correspondant aux paramètres attendus d'un code procédural externe. L'ordre de la liste *Parameters* correspondra à l'ordre des paramètres du code procédural externe. Les paramètres passés par valeur seront affectés directement à la valeur correspondante. Les paramètres passés par référence seront passés via des objets *Variable* actifs d'un type de donnée adéquat.

Description de la syntaxe:

```
Fork              -->       Target,
                            ForkSucceeded,
                            Parameters?
Target            -->       GenericObjectReference
ForkSucceeded     -->       ObjectReference
Parameters        -->        Parameter+
Parameter         -->       GenericBoolean | GenericInteger |
                            GenericOctetString |
                            GenericObjectReference |
                            GenericContentReference
```

Stop (Arrêt)

Interrompt l'exécution d'un code procédural externe.

Exécute la séquence d'actions suivante:

1) si l'objet *Program* est inactif, ne pas tenir compte de cette action: Sinon:

2) appliquer le comportement *Deactivation*.

Disposition pour l'utilisation:

- l'objet *Program* destination sera un objet *Program* disponible.

Description de la syntaxe:

```
Stop                  -->  Target
Target                -->  GenericObjectReference
```

14.5 Description formelle

```
Program Class         -->  Ingredient Class,
                           Name,
                           InitiallyAvailable?
Name                  -->  OctetString
InitiallyAvailable    -->  BOOLEAN
```

15 Classe ResidentProgram *(Programme résident)*

Description:

définit la manière de traiter les appels aux codes procéduraux exécutés localement.

Un objet *ResidentProgram* fournit une interface à un code procédural local au dispositif sur lequel fonctionne le moteur MHEG-5. La présente Recommandation ne spécifie pas un paradigme de *nommage* pour de telles interfaces et non plus la sémantique interne d'appel d'un tel programme.

classe de base:

Program

Sous-classes: aucune

Etat: classe concrète

15.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

15.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.

15.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

15.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

15.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

15.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

15.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

15.5 Description formelle

```
ResidentProgram Class  -->  Program Class
```

16 Classe *RemoteProgram* (Programme distant)

Description: définit la manière de traiter les appels aux codes procéduraux distants.

Un objet *RemoteProgram* fournit une interface à un code procédural à exécuter sur un site n'étant pas celui sur lequel fonctionne le moteur MHEG-5. L'emplacement de l'objet *RemoteProgram* est fournit grâce à l'attribut *ProgramConnectionTag*.

classe de base: *Program*

Sous-classes: aucune

Etat: classe concrète

16.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

16.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.

16.1.2 Attributs propres échangés

Cette classe définit l'attribut échangé supplémentaire suivant:

ProgramConnection-Tag

(Balise de connexion de programme)

Balise de connexion utilisée pour repérer le code procédural distant à appeler lors de l'activation de l'objet *Program*. *ProgramConnectionTag* est un identificateur de connexion ouvert par l'action *OpenConnection* de la classe Application.

ProgramConnectionTag est optionnel. Lorsqu'il n'est pas codé, le code procédural externe est repéré relativement à l'espace de *nommage* de l'application.

- Entier optionnel.
- Valeur par défaut: aucune.

16.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

16.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

16.3 Comportements internes

La sémantique du comportement interne suivant a changé par rapport à celle de la classe de base de cet objet:

Activation

(Activation)

1) si non fait pendant la préparation, rechercher l'emplacement du code procédural distant en utilisant les attributs *Name* et *ProgramConnectionTag*;
2) si le code procédural externe distant n'est pas trouvé, ne pas tenir compte de cette action. Sinon:
3) appliquer le comportement *Activation* behaviour comme hérité de la classe de base.

16.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

16.5 Description formelle

```
RemoteProgram Class     -->  Program Class,
                             ProgramConnectionTag?
ProgramConnectionTag    -->  INTEGER
```

17 Classe *InterchangedProgram* (Programme échangé)

Description: définit la manière de manipuler du code programme échangé comme le contenu *OriginalContent* d'un objet *InterchangedProgram* exécuté ou interprété sur le même dispositif que le moteur MHEG-5

La présente Recommandation ne spécifie pas comment le code programme est exécuté ou interprété dans le dispositif sur lequel fonctionne le moteur MHEG-5.

classe de base: *Program*

Sous-classes: aucune

Etat: classe concrète

17.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

17.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ou son attribut correspondant par défaut dans la classe Application (InterchangedProgramContentHook) est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.

17.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

17.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire

17.2 Evénement

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

17.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

17.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

17.5 Description formelle

```
InterchangedProgram Class    -->  Program Class
```

18 Classe Palette *(Palette)*

Description: définit une classe pour représenter une table de correspondance de couleurs

classe de base: *Ingredient*

Sous-classes: aucun

Etat: classe concrète

18.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

18.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.
InitiallyActive	*Ingredient*	Si codé, cet attribut sera mis à True pour cette classe.

18.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

18.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

18.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

18.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

18.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

18.5 Description formelle

```
Palette Class          -->  Ingredient Class
```

19 Classe *Font* (Police)

Description: définit une classe pour représenter une police de caractères utilisée pour la restitution des objets *Text*

classe de base: *Ingredient*

Sous-classes: aucun

Etat: classe concrète

19.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

19.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.
InitiallyActive	*Ingredient*	Si codé, cet attribut sera mis à True pour cette classe.

19.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

19.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

19.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

19.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

19.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

19.5 Description formelle

```
Font Class              -->  Ingredient Class
```

20 Classe *CursorShape* (Forme de curseur)

Description: définit l'encapsulation des structures de données utilisées pour représenter un curseur à déplacement libre

classe de base: *Ingredient*

Sous-classes: aucun

Etat: classe concrète

20.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

20.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.
InitiallyActive	*Ingredient*	Si codé, cet attribut sera mis à True pour cette classe.

20.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

20.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

20.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

20.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

20.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

20.5 Description formelle

```
CursorShape Class       -->  Ingredient Class
```

21 Classe Variable *(Variable)*

Description: définit une variable à l'intérieur du contexte d'un objet *Group*

classe de base: *Ingredient*

Sous-classes: *BooleanVariable, IntegerVariable, OctetStringVariable, ObjectReferenceVariable, ContentRefVariable*

Etat: classe abstraite

21.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

21.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
InitiallyActive	*Ingredient*	Si codé, cet attribut sera mis à True pour cette classe.

21.1.2 Attributs propres échangés

Cette classe définit l'attribut échangé supplémentaire suivant:

OriginalValue Valeur de la variable à la préparation.

(Valeur d'origine) L'attribut *OriginalValue* sera de l'un des types suivants: *Boolean, Integer, OctetString, ObjectReference,* ou *ContentReference.*

21.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Value Valeur courante de la variable.

(Valeur) L'attribut *Value* peut être de l'un des types suivants: *Boolean*, *Integer*, *OctetString*, *ObjectReference*, ou *ContentReference*. La seule action élémentaire pouvant modifier cet attribut est l'action *SetVariable*.

- Valeur initiale: valeur de l'attribut *OriginalValue.*

21.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, l'événement suivant est défini:

TestEvent

(Evénement test) Cet événement sera généré par le moteur MHEG-5 pour indiquer qu'une sous-classe de la classe *Variable* a été testée.

- Donnée associée: Booléen. Le résultat de la comparaison entre la variable et le paramètre de l'action *TestVariable.*

21.3 Comportements internes

La sémantique du comportement interne suivant a changé par rapport à celui de la classe de base de cet objet:

Activation

(Activation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Activation* comme défini dans sa classe de base;

2) mettre l'attribut *RunningStatus* à Vrai et générer un événement *IsRunning*.

21.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetVariable (*NewVariableValue*)

Affecte à l'attribut *Value* de l'objet *Target* la valeur *NewVariableValue*.

Affecte variable (Nouvelle valeur de variable)

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet actif de l'une des classes suivantes: BooleanVariable, IntegerVariable, OctetStringVariable, ObjectRefVariable ou ContentRefVariable;
- *NewVariableValue* contiendra ou référencera une valeur n'étant pas nécessairement du même type que l'attribut *Value* courant de la variable cible. Lorsque *NewVariableValue* et l'objet *Target* sont de types différents, *NewVariableValue* est automatiquement convertie dans le type de classe de l'objet *Target*.

NOTE – Des informations complémentaires sur les conversions sont données dans les sous-paragraphes correspondants aux sous-classes de chaque variable.

Description de la syntaxe:

```
SetVariable          -->   Target,
                           NewVariableValue
Target               -->   GenericObjectReference
NewVariableValue     -->   GenericInteger |
                           GenericBoolean |
                           GenericOctetString |
                           GenericObjectReference |
                           GenericContentReference
```

TestVariable

(*Target, Operator, ComparisonValue*)

Variable de test (Cible, Opérateur, Valeur de comparaison)

Exécute la séquence d'actions suivante:

1) comparer la valeur *Value* de la variable à celle du paramètre *ComparisonValue*. La variable *Value* est le premier opérande, le paramètre *ComparisonValue* étant le second opérande de la comparaison;

2) générer le *TestEvent* correspondant.
   - Source: cible de *TestVariable*.
   - Donnée associée: Booléen.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet actif de l'une des classes suivantes: *BooleanVariable*, *IntegerVariable*, *OctetStringVariable*, *ObjectRefVariable* ou *ContentRefVariable*;
- la valeur *ComparisonValue* sera d'un type correspondant (respectivement *GenericBoolean*, *GenericInteger*, *GenericOctetString*, *GenericObjectReference* et *GenericContentReference*). Aucune conversion implicite de type n'est autorisée;
- lorsque les valeurs sont de type *Integer*, l'opérateur sera un entier compris dans l'intervalle 1 à 6 avec la signification correspondante:

  1 signifie égal, 2 différent,

  3 strictement inférieur, 4 inférieur ou égal,

  5 strictement supérieur, 6 supérieur ou égal;
- lorsque les valeurs sont de types *Boolean*, *OctetString*, *ObjectReference* or *ContentReference*, l'opérateur sera un entier égal à 1 ou 2 avec la signification suivante:

  1 signifie égal, 2 différent.

Description de la syntaxe:

```
TestVariable          -->   Target,
                            Operator,
                            ComparisonValue
Target                -->   GenericObjectReference
ComparisonValue       -->   GenericInteger |
                            GenericBoolean |
                            GenericOctetString |
                            GenericObjectReference |
                            GenericContentReference
Operator              -->   GenericInteger
```

21.5 Description formelle

```
Variable Class      -->   Ingredient Class,
                          OriginalValue
OriginalValue       -->   BOOLEAN | INTEGER | OctetString |
                          ObjectReference | ContentReference
```

22 Classe *BooleanVariable* (Variable Booléenne)

Description: définit une variable de type Booléen à l'intérieur d'un objet *Group*

classe de base: *Variable*

Sous-classes: aucune

Etat: classe concrète

22.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

22.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, sans contraintes supplémentaires:

22.1.2 Attributs propres échangés

OriginalValue L'attribut *OriginalValue* sera un Booléen.

(Valeur d'origine)

22.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

Value

- l'attribut *Value* sera un Booléen.

(Valeur)

22.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

22.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

22.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, l'action MHEG-5 applicable suivante est définie:

SetVariable (*NewVariableValue*) Affecte variable (Nouvelle valeur de variable)	Disposition pour l'utilisation: • la nouvelle valeur *NewVariableValue* sera un *GenericBoolean*.

22.5 Description formelle

```
BooleanVariable Class     -->  Variable Class
```

23 Classe *IntegerVariable* (Variable entière)

Description: définit une variable de type Entier à l'intérieur du contexte d'un objet *Group*

classe de base: *Variable*

Sous-classes: aucune

Etat: classe concrète

23.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

23.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, sans contraintes supplémentaires:

23.1.2 Attributs propres échangés

OriginalValue L'attribut *OriginalValue* sera un Entier.

(Valeur d'origine)

23.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Value L'attribut *Value* sera un Entier.

(Valeur)

23.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

23.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

23.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetVariable (*NewVariableValue*)

Affecte variable (Nouvelle valeur de variable)

Disposition pour l'utilisation:

- la valeur *NewVariableValue* sera de l'un des deux types suivants: *Entier* ou *OctetString*.
  - lorsque *NewVariableValue* est de type *Entier*, la valeur de la variable *Target* est mise à la valeur *NewVariableValue*;
  - Lorsque *NewVariableValue* est de type *OctetString*, elle est d'abord convertie en *Entier*, et ensuite la valeur de la variable *Target* est mise à la valeur de l'entier.

    Règles de conversion:

    1) la conversion *OctetString* utilisera l'attribut *CharacterSet* de l'objet *Application*. Si plusieurs ensembles de caractères sont listés, leur conversion sera effectuée tant que leur code restera numérique;
    2) la conversion considérera que *OctetString* représente l'entier en base 10;
    3) la conversion sera faite sur les premiers caractères numériques (selon l'ISO/CEI 646: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) et s'arrêtera dès la rencontre d'un caractère non numérique [exception faite du signe moins en première position, comme détaillé dans la règle 4) ci-dessous]. Par exemple un *OctetString* représentant "123abc" sera converti en entier 123;
    4) le signe moins est autorisé comme premier caractère d'un *OctetString*. Par exemple, l'*OctetString* représentant "–123" sera converti en entier –123, mais "12-345" sera converti en entier 12 selon la règle 3).

Add (*Value*)

Ajoute (Valeur)

Ajoute la valeur *Value* à la variable *Target*. La variable *Target* est le premier opérande de l'opération en mémoire. Le résultat est stocké dans la variable *Target*.

Disposition d'utilisation:

- l'objet *Target* sera un objet *IntegerVariable* actif.

Description de la syntaxe:

```
Add                 -->   Target,
                          Value
Target              -->   GenericObjectReference
Value               -->   GenericInteger
```

Subtract (*Value*)

Soustrait (Valeur)

Soustrait la valeur *Value* de la variable *Target*. La variable *Target* est le premier opérande de l'opération en mémoire. Le résultat est stocké dans la variable *Target*.

Disposition d'utilisation:

- l'objet *Target* sera un objet *IntegerVariable* actif.

Description de la syntaxe:

```
Subtract            -->   Target,
                          Value
Target              -->   GenericObjectReference
Value               -->   GenericInteger
```

Multiply (*Value*)

Multiplie (Valeur)

Multiplie la variable *Target* par la valeur *Value*. La variable *Target* est le premier opérande de l'opération en mémoire. Le résultat est stocké dans la variable *Target*.

Disposition d'utilisation:

- l'objet *Target* sera un objet *IntegerVariable* actif.

Description de la syntaxe:

```
Multiply            -->   Target,
                          Value
Target              -->   GenericObjectReference
Value               -->   GenericInteger
```

Divide (*Value*)

Divise (Valeur)

Divise la variable *Target* par la valeur *Value*. La variable *Target* est le premier opérande de l'opération en mémoire. Lorsque le résultat n'est pas un entier, il est arrondi.

Disposition d'utilisation:

- l'objet *Target* sera un objet *IntegerVariable* actif.

Description de la syntaxe:

```
Divide              -->   Target,
                          Value
Target              -->   GenericObjectReference
Value               -->   GenericInteger
```

Modulo (*Value*)

Modulo (Valeur)

Renvoie le reste modulo *Value* de la variable *Target* selon la définition des règles arithmétiques habituelles, c'est-à-dire quelque soit les entiers a et b l'égalité suivante est satisfaite:

$$(a \text{ DIV } b) \times b + (a \text{ MOD } b) = a.$$

La variable *Target* est le premier opérande de l'opération. Le résultat est stocké dans la variable *Target*.

Disposition d'utilisation:

- l'objet *Target* sera un objet *IntegerVariable* actif.

Description de la syntaxe:

```
Modulo              -->   Target,
                          Value
Target              -->   GenericObjectReference
Value               -->   GenericInteger
```

23.5 Description formelle

```
IntegerVariable Class      -->  Variable Class
```

24 Classe *OctetStringVariable* (variable chaîne d'octets)

Description: définit une variable de type *OctetString* à l'intérieur du contexte d'un objet *Group*

classe de base: *Variable*

Sous-classes: aucun

Etat: classe concrète

24.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

24.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, sans contraintes supplémentaires:

24.1.2 Attributs propres échangés

OriginalValue (Valeur d'origine)

L'attribut *OriginalValue* sera un *OctetString*.

24.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Value
(Valeur)

- l'attribut *Value* sera un *OctetString*.

24.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

24.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

24.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetVariable (*NewVariableValue*)
Affectation (Nouvelle valeur)

Disposition pour l'utilisation:

- la valeur *NewVariableValue* sera de l'un des deux types suivants: Entier ou OctetString.
  - lorsque la valeur *NewVariableValue* est de type *OctetString*, la valeur de la variable *Target* est mise à la valeur *NewVariableValue*;
  - lorsque la valeur *NewVariableValue* est de type Entier, elle est d'abord convertie en *OctetString*, ensuite la valeur de la variable *Target* est mise à celle de la valeur *OctetString*.

    Règles de conversion:

    1) la conversion d'entier à *OctetString* utilisera l'attribut *CharacterSet* de l'objet *Application*;
    2) la conversion sera faite conformément à la représentation d'un entier en base 10.

Append (*AppendValue*)

Annexe à la variable *Target* la valeur *Value*. La variable *Target* est le premier opérande de l'opération. Le résultat est stocké dans la variable *Target*.

Accole (Valeur accolée)

Disposition d'utilisation:

- l'objet *Target* sera un objet *OctetStringVariable* actif.

Description de la syntaxe:

```
Append              -->   Target,
                          AppendValue
Target              -->   GenericObjectReference
AppendValue         -->   GenericOctetString
```

24.5 Description formelle

```
OctetStringVariable Class      -->  Variable Class
```

25 Classe *ObjectRefVariable* (Variable Référence à objet)

Description: définit une variable de type *ObjectReference* à l'intérieur du contexte d'un objet *Group*

classe de base: *Variable*

Sous-classes: aucune

Etat: classe concrète

25.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

25.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, sans contraintes supplémentaires:

25.1.2 Attributs propres échangés

OriginalValue (Valeur d'origine)

L'attribut *OriginalValue* sera un *ObjectReference*.

25.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Value (Valeur)

- l'attribut *Value* sera une *ObjectReference*.

25.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

25.3 Comportements internes

Les comportements internes de cette classe sont ceux de sa classe de base.

25.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, l'action MHEG-5 applicable suivante est définie:

SetVariable (*NewVariableValue*)

Affecte Variable (Nouvelle valeur de variable)

Disposition pour l'utilisation:

- la valeur *NewVariableValue* sera une référence *GenericObjectReference*.

25.5 Description formelle

```
ObjectRefVariable Class     --> Variable Class
```

26 Classe *ContentRefVariable* (Variable référence à contenu)

Description: définit une variable de type *ContentReference* à l'intérieur du contexte d'un objet *Group*

classe de base: *Variable*

Sous-classes: aucune

Etat: classe concrète

26.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

26.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, sans contraintes supplémentaires:

26.1.2 Attributs propres échangés

OriginalValue (Valeur d'origine)

L'attribut *OriginalValue* sera une *ContentReference.*

26.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Value (Valeur)

- l'attribut *Value* sera une *ContentReference*.

26.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

26.3 Comportements internes

Les comportements internes de cette classe sont ceux de sa classe de base.

26.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, l'action MHEG-5 applicable suivante est définie:

SetVariable (*NewVariableValue*)

Affecte Variable (Nouvelle valeur de variable)

Disposition pour l'utilisation:

- la valeur *NewVariableValue* sera une *GenericContentReference*.

26.5 Description formelle

```
ContentRefVariable Class     --> Variable Class
```

27 Classe *Presentable* (Présentable)

Description: définit le comportement des objets pouvant être présentés dans une *Scene*

classe de base: *Ingredient*

Sous-classes *Stream*, *Visible*, *Audio*, *TokenGroup*

Etat: classe abstraite

27.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

27.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base avec une sémantique identique.

27.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

27.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

27.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

27.3 Comportements internes

Les comportements internes de cette classe ont la même sémantique que ceux de sa classe de base.

27.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique, sauf pour celle mentionnée ci-dessous (*SetData*). De plus, elle définit les actions MHEG-5 applicables suivantes:

SetData (Affecte donnée)

Exécute les actions suivantes de manière synchrone:

1) exécuter l'action *SetData* action comme définie dans la classe *Ingredient*;

2) si l'objet *Presentable* est actif, le présenter à nouveau immédiatement en utilisant la nouvelle valeur de *Content*.

Les dispositions pour l'utilisation et la description de la syntaxe sont fournies dans la classe *Ingredient*.

Run

(Exécute)

Exécute les actions suivantes:

1) si le *Presentable* destination est actif, laisser tomber cette action;

2) si le *Presentable* destination est non disponible ou inactif, appliquer le comportement *Activation* du *Presentable*.

Description de la syntaxe:

```
Run                 -->  Target

Target              -->  GenericObjectReference
```

Stop

(Arrêt)

Exécute les actions suivantes:

1) si le *Presentable* destination est inactif, laisser tomber cette action;

2) si le *Presentable* destination est actif, appliquer le comportement *Deactivation* du *Presentable*.

Disposition pour l'utilisation:

- l'objet *Target* sera disponible.

Description de la syntaxe:

```
Stop                -->   Target

Target              -->   GenericObjectReference
```

27.5 Description formelle

```
Presentable Class    -->   Ingredient Class
```

28 Classe *TokenManager* (Gestionnaire de jeton)

Description: classe diverse définissant des fonctions pour gérer la navigation d'un jeton logique entre un groupe d'éléments. Le jeton peut être utilisé pour donner un comportement spécial à un élément du groupe, tel que la surbrillance dans un schéma de navigation de type "saut de surbrillance en surbrillance"

classe de base: aucune

Sous-classes: *TokenGroup*

Etat: classe abstraite

28.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

28.1.1 Attributs hérités

Cette classe ne possède pas d'attributs hérités.

28.1.2 Attributs propres échangés

Cette classe définit les attributs propres échangés suivants:

MovementTable

(Table de déplacement)

Table indiquant vers quel élément déplacer le jeton selon le premier élément et un nombre arbitraire de déplacements.

L'attribut *MovementTable* décrit une fonction discrète $c2 = f(c1, m)$, où $c1$ et $c2$ sont des indices d'éléments, et m le paramètre de déplacement utilisé dans l'action *Move*. L'index 0 indique qu'aucun élément ne détient le jeton.

Par exemple, si le jeton est sur l'élément 2 et que l'action *Move*(4) est exécutée, l'expression $f(2, 4)$ évalue le numéro de l'élément auquel le jeton doit être donné.

La fonction f est représentée comme un tableau $N \times M$ dans lequel N est le nombre d'éléments du groupe et M le nombre de déplacements possibles. La référence aux éléments se fait par un index numérique.

- Attribut optionnel.
- Séquence de structures de données *Movement*. Chaque *Movement* possède un identificateur *MovementIdentifier* implicite qui est un indice entier calculé selon la position de *Movement* dans la séquence. Chaque *Movement* comprend:
  - *TargetElements*: séquence d'entiers. Ces entiers définissent l'élément qui obtiendra le jeton lorsqu'un *Movement* est effectué.
- Valeur par défaut: aucune.

Exemple de table de déplacement:

apparence des éléments de la scène:

Elément 1 Elément 2

Elément 3 Elément 4

Contenu de la table *Movement*:

Jeton à:	Elément 1	Elément 2	Elément 3	Elément 4
Move(1)	1	2	1	2
Move(2)	3	4	3	4
Move(3)	1	1	3	3
Move(4)	2	2	4	4
Move(5)	4	2	3	4
Move(6)	1	2	2	4

Selon cette table, l'action *Move(3)* aura pour effet de donner le jeton à l'élément 1 lorsque le jeton est sur l'élément 2. L'action *Move(4)* n'aura aucun effet lorsque le jeton sera sur l'élément 4.

Les déplacements montrés dans la table pourraient correspondre à des clés de contrôle distantes comme suit:

Move(1) → Flèche vers le haut	*Move*(4) → Flèche vers la droite
Move(2) → Flèche vers le bas	*Move*(5) → Diagonale bas droite
Move(3) → Flèche vers la gauche	*Move*(6) → Diagonale haut droite

NOTE – Si une case de la table de déplacement contient la valeur 0, le *Move* correspondant aura pour effet qu'aucun élément n'aura plus le jeton. Voir *TokenPosition* plus bas.

28.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

TokenPosition
(Position de jeton)

Indice de l'élément possédant le jeton.

- Entier compris dans l'intervalle [0, nombre d'éléments].
- Valeur initiale: 1.
- La valeur 0 veut dire qu'aucun élément n'a le jeton.

28.2 Evénements

Cette classe définit les événements suivants:

TokenMovedFrom
(Jeton déplacé de)

Cet événement est généré lors de la perte du jeton par un élément.

- Donnée associée: *Index*. C'est un Entier représentant l'indice de l'élément ayant perdu le jeton.

TokenMovedTo
(Jeton déplacé vers)

Cet événement est généré lors de la récupération du jeton par un élément.

- Donnée associée: *Index*. C'est un Entier représentant l'indice de l'élément ayant récupéré le jeton.

28.3 Comportements internes

Cette classe définit le comportement interne suivant:

TransferToken (*TargetElement*)
Transfert Jeton (Elément destination)

Exécute la séquence d'actions suivante:

1) générer l'événement *TokenMovedFrom* avec la valeur de l'attribut *TokenPosition* comme donnée associée;
2) mettre l'attribut *TokenPosition* à la valeur du paramètre *TargetElement*;
3) générer l'événement *TokenMovedTo* avec la valeur de l'attribut *TokenPosition* comme donnée associée.

Les événements *TokenMovedTo* et *TokenMovedFrom* seront générés même si la valeur de *TokenPosition* avant (ou après) le départ du jeton était égale à 0.

28.4 Effet des actions MHEG-5

Cette classe définit les actions MHEG-5 applicables suivantes:

Move

(Déplace)

Déplace le jeton entre les éléments d'un groupe. Le déplacement à appliquer à partir de n'importe quel élément particulier est décrit dans l'attribut *MovementTable*.

Exécute la séquence d'actions suivante:

1) déterminer l'élément *TargetElement* en utilisant la table *MovementTable*;

2) si l'élément *TargetElement* ne possède pas encore le jeton, appliquer le comportement *TransferToken(TargetElement)* de l'objet *TokenManager*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *TokenManager* disponible.

Description de la syntaxe:

```
Move                 -->    Target,
                            Movement Identifier
Target               -->    GenericObjectReference
Movement Identifier  -->    GenericInteger
```

MoveTo (*Index*)

Déplace vers (Index)

Déplace le jeton vers un élément spécifique du groupe.

Exécute la séquence d'actions suivante:

1) déterminer l'élément *TargetElement* en utilisant le paramètre *Index* de l'action *MoveTo*;

2) si l'élément *TargetElement* ne possède pas encore le jeton, appliquer le comportement *TransferToken(TargetElement)* de l'objet *TokenManager*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *TokenManager* disponible;
- *Index* sera affecté à l'intérieur de l'intervalle [0, *N*], où *N* est le nombre d'éléments du groupe.

Description de la syntaxe:

```
MoveTo               -->    Target,
                            Index
Target               -->    GenericObjectReference
Index                -->    GenericInteger
```

GetTokenPosition (*TokenPositionVar)*

Met la variable référencée par *TokenPositionVar* à la valeur de l'attribut *TokenPosition*.

Récupère la position de jeton (Variable position de jeton)

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *TokenManager* disponible;
- *TokenPositionVar* fera référence à un objet *IntegerVariable* actif.

Description de la syntaxe:

```
GetTokenPosition     -->   Target,
                           TokenPositionVar
Target               -->   GenericObjectReference
TokenPositionVar     -->   ObjectReference
```

28.5 Description formelle

```
TokenManager Class      -->  MovementTable?
MovementTable           -->  Movement+
Movement                -->  TargetElement+
TargetElement           -->  INTEGER
```

29 Classe *TokenGroup* (Groupe de jeton)

Description: définit un groupe d'objets *Visible* en vue d'une navigation

Chaque objet *Visible* du groupe peut posséder un jeton, comme défini dans la classe *TokenManager*. Sur la base des événements générés lors de déplacements de jeton, un comportement spécial peut être implémenté (comme la surbrillance) sur l'objet *Visible* détenant le jeton.

classe de base: *Presentable*, *TokenManager*

Sous-classes: *ListGroup*

Etat: classe concrète

29.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

29.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
TokenPosition	*TokenManager*	Dans la classe TokenGroup, cet attribut prendra des valeurs comprises seulement dans l'intervalle [0,*N*], où *N* est le nombre d'objets Visible de TokenGroup. La valeur 0 signifie qu'aucun Visible ne possède le jeton.

29.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

TokenGroupItems

(Eléments de groupe de jetons)

Ensemble de *TokenGroupItems* appartenant au groupe. Chaque *TokenGroupItem* contient une référence à un objet *Visible* et un nombre d'actions, appelées *ActionSlots*. Ces actions peuvent être exécutées à la demande par l'action *CallActionSlot*.

- Séquence de structures de données suivante:
  - *AVisible*: référence à un objet *Visible*;
  - *ActionSlots*: séquence optionnelle d'objets *Action*.

NoTokenActionSlots

ActionSlot pouvant être exécutée à la demande par l'action *CallActionSlot* action lorsque aucun élément ne possède le jeton.

- Attribut optionnel.
- Séquence de *ActionSlot*.

29.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

29.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

29.3 Comportements internes

Les sémantiques des comportements internes suivants de cette classe ont changé:

Activation

(Activation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Activation* comme hérité de la classe *Presentable*;
2) appliquer le comportement *Activation* à chaque élément selon l'ordre donné par TokenGroupItems;
3) générer l'événement *TokenMovedTo* avec la valeur de l'attribut *TokenPosition* comme donnée associée;
4) mettre *RunningStatus* à *True* et génère l'événement *IsRunning*.

Deactivation

(Désactivation)

1) générer l'événement *TokenMovedFrom* avec la valeur de l'attribut *TokenPosition* comme donnée associée;
2) appliquer le comportement *Deactivation* comme hérité de la classe *Presentable*.

29.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique, sauf pour ce qui est redéfini ci-dessous. De plus, l'action MHEG-5 applicable suivante est définie:

CallActionSlot (*Index*)

Appel d'action liée à la fenêtre (Index)

Exécute un objet *Action* associé à l'élément possédant le jeton.

Exécute la séquence d'actions suivante:

1) si *TokenPosition* vaut 0, considérer la séquence *ActionSlots* de l'attribut *NoTokenActionSlots*;

2) si *TokenPosition* est différent de 0, considère la séquence *ActionSlots* associée à l'élément ayant actuellement le jeton;

3) considérer l'action *ActionSlot* situé à la position *Index* dans la séquence d'*ActionSlots*. Si cet *ActionSlot* est *NULL*, alors ignorer l'effet de l'action *CallActionSlot*. Sinon, exécuter cet *ActionSlot*.

L'exécution de l'action *CallActionSlot* n'est pas terminée tant que l'action *ActionSlot* n'a pas été exécutée.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *TokenGroup* actif;
- *Index* sera affecté dans l'intervalle [0, nombre d'éléments dans *TokenGroupItem*].

Description de la syntaxe:

```
CallActionSlot       -->   Target,
                           Index
Target               -->   GenericObjectReference
Index                -->   GenericInteger
```

29.5 Description formelle

```
TokenGroup Class     -->  Presentable Class,
                          TokenManager Class,
                          TokenGroupItems,
                          NoTokenActionSlots?
NoTokenActionSlots   -->  ActionSlots
TokenGroupItems      -->  TokenGroupItem+
TokenGroupItem       -->  Avisible,
                          ActionSlots?
Avisible             -->  ObjectReference
ActionSlots          -->  ActionSlot+
ActionSlot           -->  Action Class | NULL
```

30 Classe *ListGroup* (Groupe de listes)

Description: cette classe définit les emplacements à l'écran de chaque position définie dans la table *MovementTable* héritée de la classe de base. De plus, elle définit les fonctionnalités pour présenter les objets *Visible* contenus dans une liste interne associée à ces emplacements et pour ajouter ou enlever des objets *Visibles* de cette liste interne

classe de base: *TokenGroup*

Sous-classes: aucune

Etat: classe concrète

NOTE 1 – *SetPosition, GetPosition, Run, Stop* ainsi que d'autres actions élémentaires peuvent être adressées à des objets *Visibles* de *ListGroup*. Cependant l'auteur devrait avertir des effets de bord possibles de telles actions (plus spécialement pour ce qui concerne les *Positions* des objets *Visibles*).

NOTE 2 – Bien que ce ne soit pas recommandé, il est permis d'avoir un objet *Visible* référencé par plusieurs *ListGroups*. Dans un tel cas, les comportements de cet objet *Visible* dépendront du *ListGroup* ou des actions élémentaires qui sont invoquées. L'auteur devrait être conscient des effets de bord possibles.

30.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

30.1.1 Attributs hérités

Cette classe possède les mêmes attributs que sa classe de base.

30.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Positions
(Positions)

Ensemble de coordonnées d'écran associées aux positions spécifiées par l'attribut *TokenMovementTable* hérité de *TokenGroup*. Ces éléments spécifient tous ensemble les cellules de *ListGroup*. Chaque cellule possède un index, identique à la position logique définie dans la table *TokenMovementTable*.

- Séquence de la structure de donnée suivante:
  - Position: Paire d'entiers (*XPosition*, *Yposition*).

WrapAround
(Défilement bouclé)

Un booléen optionnel indiquant le comportement du *ListGroup* pour ce qui concerne la présentation des éléments de la liste *ItemList*.

- Booléen optionnel;
- valeur par défaut: *False*.

MultipleSelection
(Sélection multiple)

Un booléen optionnel déterminant si le *ListGroup* permet à des éléments multiples d'avoir simultanément leur *ItemSelectionStatus* mis à *True*.

- Booléen optionnel;
- valeur par défaut: *False*.

30.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

ItemList

(Liste d'éléments)

Ensemble des références aux objets *Visibles* appartenant au Groupe. Chaque référence possède implicitement un numéro d'index , à partir de 1. Ce numéro d'index est utilisé pour faire référence aux références dans l'attribut *ItemList*.

Chaque élément dans la liste *ItemList* possède un attribut *ItemSelectionStatus*. Cet *ItemSelectionStatus* est un booléen affecté par défaut à la valeur *False*. Cet attribut peut être affecté en utilisant les actions élémentaires *SelectItem* et *DeselectItem*; sa valeur peut être renvoyée au moyen de l'action élémentaire *GetItemStatus*. Il faut noter que cet attribut ne fait pas partie des objets *Visibles* référencés à partir de la liste *ItemList*.

Lorsque *ListGroup* est actif, les éléments de *ItemList* sont présentés aux cellules à partir de l'élément indiqué par l'attribut *FirstItem*. Les autres éléments sont rendus inactifs. La présentation dépend aussi de la valeur de l'attribut *WrapAround*.

Lors de la préparation, cet ensemble est initialisé à l'ensemble des références listées dans l'attribut *TokenGroupItems*.

- Ensemble des références aux objets *Visibles*
- Valeur initiale: ensemble vide

FirstItem

(Premier élément)

L'indice de l'élément de la liste *ItemList* qui est présenté à la première cellule. Ceci définit une fenêtre sur la liste ordonnée des éléments de *ItemList*. Cette fenêtre est égale en taille au nombre des cellules, et sa position au regard des éléments peut être changée au moyen de l'action élémentaire *ScrollItems*.

La présentation des éléments dans la liste dépend de la position de cette fenêtre et de la valeur de l'attribut *WrapAround*.

S'il y a au moins autant d'éléments dans la liste *ItemList* que de cellules, la fenêtre peut être facilement visualisée. Cette situation est décrite deux fois ci-après. Dans le premier exemple, *WrapAround* est *False*; c'est-à-dire qu'il n'y a pas de bouclage de défilement.

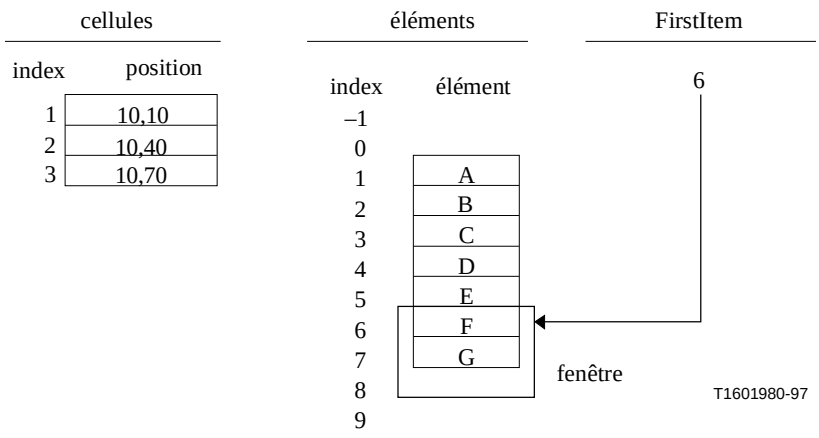


Figure 6/T.172 – Présentation lorsque *WrapAround* est *False*

Dans ce cas, les éléments F et G sont présentés respectivement aux cellules 1 et 2 tandis que les autres éléments ne le sont pas. La cellule 3 reste inutilisée.

Dans l'exemple qui suit, *WrapAround* vaut *True*; c'est-à-dire qu'il n'y a pas de bouclage en défilement.

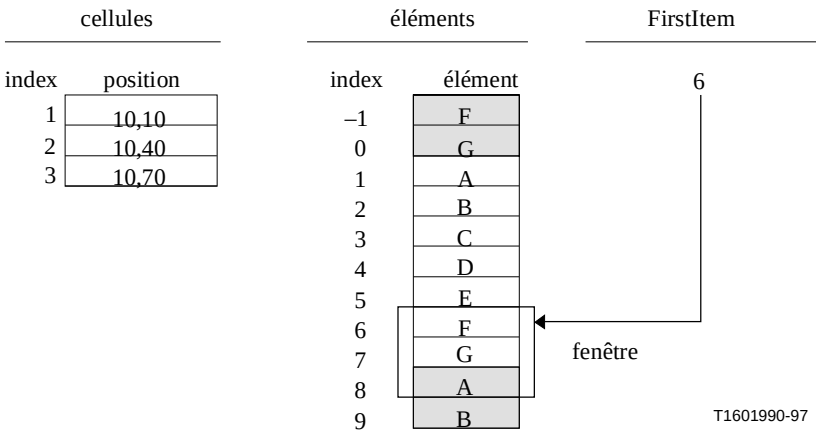


Figure 7/T.172 – Présentation lorsque *WrapAround* vaut *True*

Dans ce cas, les éléments F, G et A sont présentés respectivement aux cellules 1, 2 et 3 tandis que les autres éléments ne le sont pas. Il faut noter que dans une "wrapping liste" les indices des éléments sont étendus (modulo le nombre d'éléments de la liste). Par exemple,...,–4, 3, 10,... sont tous des indices valides pour l'élément C dans l'exemple ci-dessus.

L'autre situation apparaît s'il y a moins d'éléments dans *ItemList* que de cellules. Cette situation est décrite deux fois ci-après. Dans l'exemple qui suit, *WrapAround* est *False*; c'est-à-dire qu'il n'y a pas de bouclage en défilement.

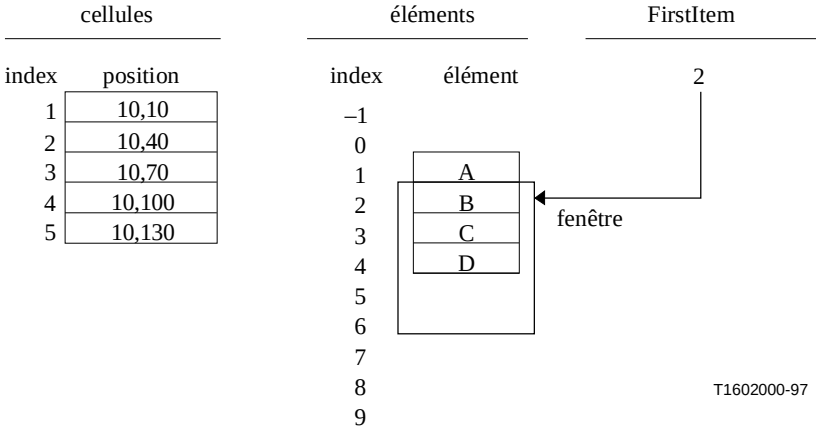


Figure 8/T.172 – Présentation de quelques éléments lorsque WrapAround vaut False

Ce cas ressemble plutôt au premier exemple, seule la fenêtre est plus grande. Les éléments B, C et D sont présentés respectivement aux cellules 1, 2 et 3 tandis que l'élément A ne l'est pas. Les cellules 4 et 5 restent inutilisées.

Dans l'exemple suivant , *WrapAround* vaut *True*; c'est-à-dire qu'il n'y a pas de bouclage en défilement.

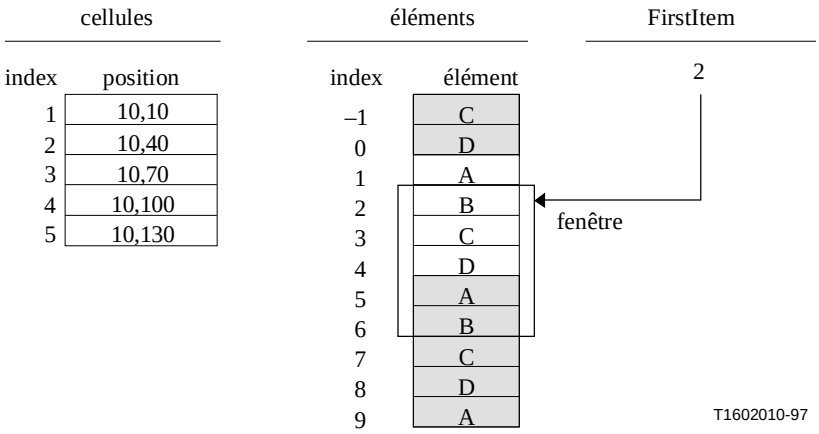


Figure 9/T.172 – Présentation de quelques éléments lorsque *WrapAround* vaut *True*

La chose importante dans ce cas est qu'il y a deux cellules candidates à être présentées à l'élément B. Comme les objets *Visibles* ne peuvent avoir qu'une seule position, les éléments peuvent aussi être présentés dans une seule cellule. Si selon la fenêtre, un élément peut être présenté à plus d'une cellule, il le sera seulement à la cellule d'indice le plus faible. Dans ce cas, l'élément B est présenté à la cellule 1. Les éléments C, D et A sont présentés respectivement aux cellules 2, 3 et 4. La cellule 5 reste inutilisée. Il faut noter que lorsque le nombre d'éléments est inférieur au nombre de cellules, certaines cellules resteront inutilisées même si c'est une "wrapping liste".

Enfin, le *FirstItem* peut accepter toute valeur entière. Dans certains cas, lorsque *WrapAround* est *False*, il peut s'avérer que chaque cellule soit vide ; un tel cas est autorisé lorsque aucun objet *Visible* n'est affiché.

- Entier.
- Valeur initiale: 1.

30.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, les événements suivants sont définis:

FirstItemPresented

(Premier élément présenté)

Cet événement est généré à chaque fois que change l'état de présentation du premier élément dans la liste *ItemList*. La valeur associée reflétera le nouvel état de présentation. L'état de présentation de l'élément change si l'attribut *FirstItem* a changé ou si le numéro d'éléments dans la liste est changé.

- Donnée associée, Booléen: *True* si l'élément est présenté, *False* sinon.

LastItemPresented

(Dernier élément présenté)

Cet événement est généré à chaque fois que change l'état de présentation du premier élément dans la liste *ItemList*. La valeur associée reflétera le nouvel état de présentation. L'état de présentation de l'élément change si l'attribut *FirstItem* a changé ou si le numéro d'éléments dans la liste est changé.

- Donnée associée, Booléen: *True* si l'élément est présenté, *False* sinon.

HeadItems

(Eléments de tête)

Cet événement est généré à chaque fois que change le numéro d'élément dans *ItemList* avec un indice compris dans l'intervalle [1, *FirstItem* –1]. Ce numéro peut changer si l'attribut *FirstItem* a changé ou si le numéro d'éléments dans la liste est changé.

- Donnée associée, Entier: le nombre d'éléments dans *ItemList* avec un indice plus petit que *FirstItem*.

TailItems

(Eléments de queue)

Cet événement est généré à chaque fois que change le numéro d'élément dans *ItemList* avec un indice compris dans l'intervalle *FirstItem*, nombre d'éléments. Ce numéro peut changer si l'attribut *FirstItem* a changé ou si le numéro d'éléments dans la liste est changé.

- Donnée associée, Entier: le nombre d'éléments dans *ItemList* avec un indice supérieur ou égal à *FirstItem*.

ItemSelected

(Elément sélectionné)

Cet événement est généré si *ItemSelectionStatus* d'un élément est changé à *True*.

- Donnée associée, Entier: l'indice de l'élément.

ItemDeselected

(Elément désélectionné)

Cet événement est généré si *ItemSelectionStatus* d'un élément est changé à *True*.

- Donnée associée, Entier: l'indice de l'élément.

30.3 Comportements internes

Cette classe définit les comportements internes suivants:

Preparation

(Préparation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Preparation* comme hérité de sa classe de base;
2) ajouter chaque référence listée dans l'attribut *TokenGroupItems* à la liste *ItemList*. Si un objet *Visible* est référencé plus d'une fois dans *TokenGroupItems*, il est ajouté seulement une fois dans *ItemList*.

Destruction

(Destruction)

Exécute la séquence d'actions suivante:

1) remettre tous les objets *Visibles* de *ListGroup* à leur *OriginalPosition*;
2) appliquer le comportement *Destruction* comme hérité de la classe de base.

Activation
(Activation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Activation* comme hérité de sa classe de base;
2) appliquer le comportement *Update*.

Deactivation
(Désactivation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Deactivation* à tous les objets *Visibles* référencés dans *ItemList*;
2) appliquer le comportement *Deactivation* comme défini dans la classe de base.

Update
(Mise à jour)

Exécute la séquence d'actions suivante:

1) pour chaque élément devant être présenté à une cellule donnée, mettre sa *Position* (attribut interne) à la position définie pour cette cellule. Par la suite, si le *RunningStatus* de *ListGroup* est à *True* et l'élément lui-même inactif, lui appliquer le comportement *Activation*;
2) pour chaque élément n'ayant pas à être présenté, mettre sa position à celle de son attribut *OriginalPosition*. Par la suite , si le *RunningStatus* de *ListGroup* est à *True* et l'élément actif, lui applique le comportement *Deactivation*.

Additem
(*Index, Item*)
Ajout d'élément
(Indice, élément)

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* référencé par le paramètre *Item* est déjà dans *ItemList*, ignorer ce comportement;
2) si *Index* est inférieur à 1 ou supérieur au numéro courant de *ItemList* +1, ignorer ce comportement;
3) ajouter la référence à l'élément à la position *Index* dans *ItemList*. (Les éléments occupant précédemment la position *Index* dans *ItemList* – s'ils existent – auront maintenant un indice égal à *Index* +1 et, de manière identique, chaque élément d'indice supérieur à *Index* aura maintenant un indice augmenté de 1);
4) appliquer le comportement *Update*.

Delitem
(*Item*)
Destruction d'élément
(Elément)

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* référencé par le paramètre *Item* n'est pas dans *ItemList*, ignorer ce comportement;
2) enlever la référence indiquée par le paramètre *Item* de *ItemList*. (L'indice d'élément des éléments suivants dans *ItemList* sera décrémenté de 1);
3) mettre la position de l'objet *Visible* référencé à son *OriginalPosition*;
4) appliquer le comportement *Update*.

Select (*ItemIndex*)

Sélection (Indice d'élément)

Exécute la séquence d'actions suivante:

1) si l'état *ItemSelectionStatus* de l'élément d'indice *ItemIndex* est à *True*, ignorer ce comportement;
2) si *MultipleSelection* est à *False*, appliquer le comportement interne *Deselect(Index)* à tout élément d'indice *Index*, pour lequel *ItemSelectionStatus* est à *True*;
3) mettre l'état *ItemSelectionStatus* de l'élément d'indice *ItemIndex* à *True*;
4) générer l'événement *ItemSelected* avec *ItemIndex* comme donnée associée.

Deselect (*ItemIndex*)

Désélection (Indice d'élément)

Exécute la séquence d'actions suivante:

1) si l'état *ItemSelectionStatus* de l'élément d'indice *Index* est à *False*, ignorer ce comportement;
2) mettre l'état *ItemSelectionStatus* de l'élément d'indice *Index* à *False*;
3) générer l'événement *ItemDeselected* avec *Index* comme donnée associée.

30.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

AddItem (*ItemIndex*, *VisibleReference*)

Ajoute élément (Indice d'élément, Référence à objet Visible)

Applique le comportement interne *Additem(ItemIndex*, *VisibleReference)* pour rajouter la référence à l'objet *Visible*, comme indiqué par le paramètre *VisibleReference* à la position indiquée par le paramètre *ItemIndex*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible;
- *VisibleReference* fera référence à un objet *Visible* disponible, n'étant référencé par aucune référence dans *ItemList*.

Description de la syntaxe:

```
AddItem              -->    Target,
                            ItemIndex,
                            VisibleReference
Target               -->    GenericObjectReference
ItemIndex            -->    GenericInteger
VisibleReference     -->    GenericObjectReference
```

DelItem (*VisibleReference*)

Détruit élément (Référence à objet Visible)

Applique le comportement interne *Delitem(VisibleReference)* pour détruire la référence à l'objet *Visible*, comme indiqué par le paramètre *VisibleReference* de l'attribut *ItemList*, s'il apparaît dedans.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible.

Description de la syntaxe:

```
DelItem              -->    Target,
                            VisibleReference
Target               -->    GenericObjectReference
VisibleReference     -->    GenericObjectReference
```

GetListItem (*ItemIndex, ItemRefVar*)

Récupère élément de liste (Indice d'élément, Variable de référence à élément)

Renvoie la référence incluse dans l'attribut *ItemList* à l'indice spécifié par le paramètre *ItemIndex* de *ObjectRefVariable* référencée par *ItemRefVar*.

Si *WrapAround* est à *False*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, ignorer cette action.

Si *WrapAround* est à *True*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, *ItemIndex* sera interprété comme le modulo du nombre d'éléments dans l'attribut *ItemList*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible;
- *ItemRefVar* fera référence à un objet *ObjectRefVariable* actif;
- si *WrapAround* est à *False*, *ItemIndex* sera compris dans l'intervalle [1, nombre d'éléments].

Description de la syntaxe:

```
GetListItem          -->    Target,
                            ItemIndex,
                            ItemRefVar
Target               -->    GenericObjectReference
ItemIndex            -->    GenericInteger
ItemRefVar           -->    ObjectReference
```

GetCellItem (*CellIndex, ItemRefVar*)

Renvoie la référence de l'objet *Visible* présenté à la cellule indiquée dans le paramètre *CellIndex* de la variable *ObjectRefVariable* référencée par *ItemRefVar*.

Récupère élément de cellule
(Indice de cellule, Variable référence à élément)

Si *CellIndex* spécifie un indice inférieur ou égal à 1, renvoyer la référence de l'objet *Visible* présenté à la première cellule. Si *CellIndex* spécifie un indice supérieur ou égal au nombre de cellules, renvoyer la référence de l'objet *Visible* présenté à la dernière cellule. S'il n'y a pas d'objet *Visible* présenté à la cellule indiquée, renvoyer *NULL*.

NOTE – l'action *GetTokenPosition* peut être utilisée pour obtenir l'indice de la cellule détenant actuellement le jeton, comme défini dans la classe de base.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible;
- *ItemRefVar* fera référence à un objet *ObjectRefVariable* actif.

Description de la syntaxe:

```
GetCellItem          -->    Target,
                            CellIndex,
                            ItemRefVar
Target               -->    GenericObjectReference
CellIndex            -->    GenericInteger
ItemRefVar           -->    ObjectReference
```

GetItemStatus (*ItemIndex*, *ItemStatusVar*)

Renvoie la valeur de l'attribut *ItemSelectionStatus* de l'élément de *ItemList* à l'indice *ItemIndex* dans la variable *BooleanVariable* référencée par *ItemStatusVar*.

Récupère l'état de l'élément
(Indice d'élément, Variable à état d'élément)

Si *WrapAround* est à *False*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, ignorer cette action.

Si *WrapAround* est à *True*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, *ItemIndex* sera interprété comme le modulo du nombre d'éléments dans l'attribut *ItemList*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible;
- *ItemRefVar* fera référence à un objet *BooleanVariable* actif disponible;
- si *WrapAround* est à *False*, *ItemIndex* sera compris dans l'intervalle [1, nombre d'éléments].

Description de la syntaxe:

```
GetItemStatus        -->    Target,
                            ItemIndex,
                            ItemStatusVar
Target               -->    GenericObjectReference
ItemIndex            -->    GenericInteger
ItemStatusVar        -->    ObjectReference
```

SelectItem (*ItemIndex*)

Sélectionne élément (Indice d'élément)

Si *WrapAround* est à *False*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, ignorer cette action.

Si *WrapAround* est à *True*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, *ItemIndex* sera interprété comme le modulo du nombre d'éléments dans l'attribut *ItemList*.

Applique le comportement interne *Select(ItemIndex)*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible.

Description de la syntaxe:

```
SelectItem           -->    Target,
                            ItemIndex
Target               -->    GenericObjectReference
ItemIndex            -->    GenericInteger
```

DeselectItem (*ItemIndex*)

Désélectionne élément (Indice d'élément)

Si *WrapAround* est à *False*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, ignorer cette action. Sinon:

si *WrapAround* est à *True*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, *ItemIndex* sera interprété comme le modulo du nombre d'éléments dans l'attribut *ItemList*.

Applique le comportement interne *Deselect(ItemIndex)*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible.

Description de la syntaxe:

```
DeselectItem         -->    Target,
                            ItemIndex
Target               -->    GenericObjectReference
ItemIndex            -->    GenericInteger
```

ToggleItem (*ItemIndex*)

Inverse sélection (Indice d'élément)

Si *WrapAround* est à *False*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, ignorer cette action. Sinon:

si *WrapAround* est à *True*, et si *ItemIndex* spécifie un indice inférieur à 1 ou plus grand que le nombre d'éléments dans l'attribut *ItemList*, *ItemIndex* sera interprété comme le modulo du nombre d'éléments dans l'attribut *ItemList*;

si l'état *ItemSelectionStatus* de l'élément indiqué par *ItemIndex* est à *True*, applique le comportement interne *Deselect(ItemIndex)*, sinon appliquer le comportement interne *Select(ItemIndex)*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible.

Description de la syntaxe:

```
ToggleItem          -->    Target,
                           ItemIndex,
Target              -->    GenericObjectReference
ItemIndex           -->    GenericInteger
```

ScrollItems (*ItemsToScroll*)

Déroule les éléments (Eléments à dérouler)

Ajoute *ItemsToScroll* à l'attribut *FirstItem*, et applique le comportement *Update*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible.

Description de la syntaxe:

```
ScrollItems         -->    Target,
                           ItemsToScroll
Target              -->    GenericObjectReference
ItemsToScroll       -->    GenericInteger
```

SetFirstItem (*NewFirstItem*)

Affecte le premier élément (Nouveau premier élément)

Met la valeur de l'attribut *FirstItem* à *NewFirstItem*. Applique le comportement *Update*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible.

Description de la syntaxe:

```
SetFirstItem        -->    Target,
                           NewFirstItem
Target              -->    GenericObjectReference
NewFirstItem        -->    GenericInteger
```

GetFirstItem (*FirstItemVar*)

Récupère le premier élément (Variable à premier élément)

Renvoie la valeur courante de l'attribut *FirstItem* dans la variable *IntegerVariable* référencée par *FirstItemVar*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible;
- *FirstItemVar* fera référence à un objet *IntegerVariable* disponible.

Description de la syntaxe:

```
GetFirstItem          -->    Target,
                             FirstItemVar
Target                -->    GenericObjectReference
FirstItemVar          -->    ObjectReference
```

GetListSize (*SizeVar*)

Récupère la taille de liste (Référence à taille)

Renvoie la nombre d'éléments de *ItemList* dans la variable *IntegerVariable* référencée par *SizeVar*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *ListGroup* disponible;
- *SizeVar* fera référence à un objet *IntegerVariable* disponible.

Description de la syntaxe:

```
GetListSize           -->    Target,
                             SizeVar
Target                -->    GenericObjectReference
SizeVar               -->    ObjectReference
```

30.5 Description formelle

```
ListGroup Class       -->    TokenGroup Class,
                             Positions,
                             WrapAround?,
                             MultipleSelection?
Positions             -->    Position*
Position              -->    XYposition,
WrapAround            -->    BOOLEAN□
MultipleSelection     -->    BOOLEAN□
```

31 Classe *Visible* (Visible)

Description:	définit le comportement des objets *Presentables* ayant une représentation visuelle sur l'écran
classe de base:	*Presentable*
Sous-classes:	*Video*, *RTGraphics*, *Bitmap*, *LineArt*, *Text*, *Slider*, *Button*
Etat:	classe abstraite

31.1 Attributs

Le présent sous-paragraphe définit les attributs internes, échangés et hérités de cette classe.

31.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base avec une sémantique identique.

31.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

OriginalBoxSize

(Taille de l'espace d'origine)

Taille à l'origine de l'espace de délimitation de l'objet *Visible* en relation avec le système de coordonnées de l'objet *Scene*. La taille est donnée par un couple d'entiers représentant respectivement la largeur (selon l'axe des abscisses) et la hauteur (selon l'axe des ordonnées) de l'espace.

On n'utilisera pas de valeurs négatives. Les valeurs dépassant le système de coordonnées sont autorisées ; dans ce cas seule la partie de l'objet *Visible* à l'intérieur du système de coordonnées de la *Scene* sera affichée.

- Couple d'entiers (Taille espace en X, Taille espace en Y).

OriginalPosition

(Position à l'origine)

Position à l'origine du coin haut gauche de l'objet *Visible* en relation avec le système de coordonnées de la *Scene*.

Ce couple de coordonnées (X,Y) exprime la position à l'origine du coin haut gauche de l'objet *Visible* dans le système de coordonnées de la *Scene* courante active.

Les valeurs négatives et les valeurs dépassant le système de coordonnées de la *Scene* sont autorisées ; dans ce cas seule la partie de l'objet *Visible* à l'intérieur du système de coordonnées de la *Scene* sera affichée.

- Couple d'entiers optionnel (*XPosition*, *YPosition*).
- Valeur par défaut: (0,0).

OriginalPaletteRef

(Référence à la palette d'origine)

Indique l'objet *Palette* d'origine à utiliser pour afficher les couleurs de l'objet *Visible*.

- référence optionnelle à l'objet *Palette*.
- Valeur par défaut: pas de palette.

Cet attribut est obligatoire lorsque d'autres attributs de couleurs sont affectés et nécessite des références à *Palette*.

31.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

BoxSize

(Taille de l'espace)

Taille de l'espace de délimitation de l'objet *Visible* en relation avec le système de coordonnées de la *Scene*. Les parties de l'objet *Visible* tombant en dehors des frontières de l'espace d'affichage ne seront pas affichées. Si l'objet *Visible* ne recouvre pas complètement les contenus de l'espace d'affichage, alors la partie restante sera transparente.

- Couple d'entiers (Taille espace en X, Taille espace en Y).
- Valeur initiale: valeur de l'attribut *OriginalBoxSize*.

Position

(Position)

Position du coin haut gauche de l'objet *Visible* en relation avec le système de coordonnées de la *Scene*. L'objet *Visible*, lorsqu'il est affiché, aura son coin haut gauche à cette position.

- Couple d'entiers (Taille espace en X, Taille espace en Y).
- Valeur initiale: valeur de l'attribut *OriginalPositionAttribute*.

PaletteRef

(Référence à palette)

Lors de l'utilisation d'une palette, référence l'objet *Palette* à utiliser pour afficher les couleurs de l'objet *Visible*.

L'attribut *Palette* ne sera pas défini pour les classes spécifiant que *OriginalPaletteRef* ne sera pas codé.

- référence optionnelle à un objet *Palette*.
- Valeur initiale: valeur de l'attribut *OriginalPaletteRef*.

31.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

31.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec les changements suivants dans la sémantique:

Preparation

(Préparation)

Exécute la séquence d'actions suivante:

1) exécuter les étapes 1), 2) et 3) du comportement *Preparation* comme défini dans la classe *Root*;
2) si l'objet *Visible* n'est pas référencé dans la pile *DisplayStack* de l'objet *Application* actif, ajouter une référence à cet objet *Visible* au sommet de *DisplayStack*;
3) exécuter les étapes 4), 5) et 6) du comportement *Preparation* comme défini dans la classe *Root*.

Destruction

(Destruction)

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* est référencé dans la pile *DisplayStack* de l'objet *Application* actif, enlever la référence à cet objet *Visible* du sommet de *DisplayStack*;

2) exécuter le comportement *Destruction* comme défini dans la classe de base.

Activation (Activation)

Exécute la séquence d'actions suivante:

1) exécuter le comportement *Activation* comme défini dans la classe de base;
2) afficher l'objet *Visible* selon sa position dans la pile *DisplayStack* et selon la position de son espace d'affichage défini par les attributs *Position* et *BoxSize*;
3) mettre le *RunningStatus* à *True* et générer l'événement *IsRunning*.

Deactivation (Désactivation)

Exécute la séquence d'actions suivante:

1) si l'attribut *RunningStatus* de l'objet est à *False*, ignorer ce comportement, sinon:
2) arrêter l'affichage de l'objet *Visible*;
3) exécuter le comportement *Deactivation* comme défini dans la classe de base.

31.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetPosition (*NewXPosition*, *NewYPosition*)

Affecte position (Nouvelle position X, Nouvelle position Y)

Change l'emplacement de l'objet visible destination.

Exécute la séquence d'actions suivante:

1) affecter l'attribut *Position* selon *NewXPosition* et *NewYPosition*;
2) si l'objet *Visible* est actif, redessiner l'entité graphique représentant l'objet sur l'écran dans l'espace de délimitation défini par les attributs *BoxSize* et *Position* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *NewXPosition* et *NewYPosition* correspondent à un emplacement interprété dans le système de coordonnées de la *Scene* défini par l'attribut *SceneCoordinateSystem* de l'objet *Scene* courant actif.

Description de la syntaxe:

```
SetPosition          -->   Target,
                           NewXPosition,
                           NewYPosition
Target               -->   GenericObjectReference
NewXPosition         -->   GenericInteger
NewYPosition         -->   GenericInteger
```

GetPosition (*XPositionVar*, *YPositionVar*)

Récupère la position (Variable positionX, Variable positionY)

Renvoie la position de l'objet *Scene* actif.

Met les *Variables* référencées par *XPositionVar* et *YPositionVar* à la valeur des positions X et Y respectivement de l'objet *Visible* destination.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *XPositionVar* et *YPositionVar* feront référence à des objets *IntegerVariable* actifs.

Description de la syntaxe:

```
GetPosition          -->  Target,
                          XPositionVar,
                          YPositionVar
Target               -->  GenericObjectReference
XPositionVar         -->  ObjectReference
YPositionVar         -->  ObjectReference
```

SetBoxSize (*XNewBoxSize*, *YNewBoxSize*)

Affecte la taille de l'espace (Nouvelle taille en X, Nouvelle taille en Y)

Change la taille de l'espace de délimitation de l'objet *Visible* destination.

Exécute la séquence d'actions suivante:

1) affecter l'attribut *BoxSize*;
2) redessiner l'entité graphique représentant l'objet sur l'écran dans l'espace de délimitation défini par les attributs *BoxSize* et *Position* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *XNewBoxSize* et *YNewBoxSize* seront des valeurs strictement positives.

Description de la syntaxe:

```
SetBoxSize           -->   Target,
                           XNewBoxSize,
                           YNewBoxSize
Target               -->   GenericObjectReference
XNewBoxSize          -->   GenericInteger
YNewBoxSize          -->   GenericInteger
```

GetBoxSize (*XBoxSizeVar, YBoxSizeVar*)

Récupère la taille de l'espace (Variable taille d'espace en X, Variable taille d'espace en Y)

Renvoie la taille de l'espace de délimitation de l'objet *Visible* destination.

Met les *Variables* référencées par *XBoxSizeVar* et *YBoxSizeVar* à la valeur des positions X et Y respectivement de l'objet destination.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *XBoxSizeVar* et *YBoxSizeVar* feront référence à des objets *IntegerVariable* actifs.

Description de la syntaxe:

```
GetBoxSize          -->  Target,
                         XBoxSizeVar,
                         YBoxSizeVar
Target              -->  GenericObjectReference
XBoxSizeVar         -->  ObjectReference
YBoxSizeVar         -->  ObjectReference
```

BringToFront

(Met au premier plan)

Met un objet *Visible* au premier plan sur l'écran.

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* destination n'est pas référencé dans la pile *DisplayStack* de l'objet *Application* actif, ignorer cette action;
2) l'objet *Visible* est référencé dans la pile *DisplayStack* de l'objet *Application* actif, déplacer la référence à cet objet *Visible* au sommet de *DisplayStack*;
3) l'objet *Visible* destination est actif, redessiner l'entité graphique représentant l'objet sur l'écran dans l'espace de délimitation défini par les attributs *BoxSize* et *Position* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible.

Description de la syntaxe:

```
BringToFront        -->   Target
Target              -->   GenericObjectReference
```

SendToBack

(Envoie à l'arrière-plan)

Met un objet *Visible* à l'arrière-plan de l'écran.

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* destination n'est pas référencé dans la pile *DisplayStack* de l'objet *Application* actif, ignorer cette action;

2) si l'objet *Visible* est référencé dans la pile *DisplayStack* de l'objet *Application* actif, déplacer la référence à cet objet *Visible* au bas de *DisplayStack*;

3) si l'objet *Visible* destination est actif, redessiner l'entité graphique représentant l'objet sur l'écran dans l'espace de délimitation défini par les attributs *BoxSize* et *Position* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible.

Description de la syntaxe:

```
SendToBack          -->   Target

Target              -->   GenericObjectReference
```

PutBefore (*ReferenceVisible*)

Met devant (Référence à Visible)

Met un objet *Visible* juste devant un autre dans la pile d'affichage.

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* destination n'est pas référencé dans la pile *DisplayStack* de l'objet *Application* actif, ignorer cette action;

2) si l'objet *Visible* est référencé dans la pile *DisplayStack* de l'objet *Application* actif, déplacer la référence à cet objet *Visible* à la position se situant juste devant celle de l'objet référencé par *ReferenceVisible* dans la *DisplayStack*;

3) si l'objet *Visible* destination est actif, redessiner l'entité graphique représentant l'objet sur l'écran dans l'espace de délimitation défini par les attributs *BoxSize* et *Position* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

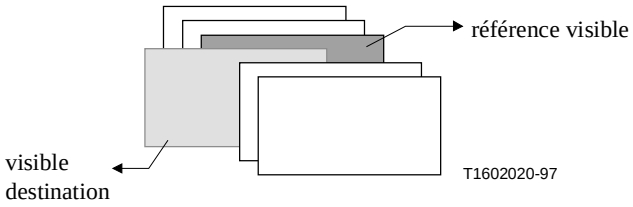


Figure 10/T.172 – Effet de l'action PutBefore

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *ReferenceVisible* sera une référence à un objet *Visible* disponible;

- la pile *DisplayStack* de l'objet *Application* actif contiendra une référence à l'objet *ReferenceVisible*.

Description de la syntaxe:

```
PutBefore            -->   Target,
                           ReferenceVisible
Target               -->   GenericObjectReference
ReferenceVisible     -->   GenericObjectReference
```

PutBehind (*ReferenceVisible*)

Met derrière (Référence à Visible)

Met un objet *Visible* juste derrière un autre dans la pile d'affichage.

Exécute la séquence d'actions suivante:

1) si l'objet *Visible* destination n'est pas référencé dans la pile *DisplayStack* de l'objet *Application* actif, ignorer cette action;
2) si l'objet *Visible* est référencé dans la pile *DisplayStack* de l'objet *Application* actif, déplacer la référence à cet objet *Visible* à la position se situant juste après celle de l'objet référencé par *ReferenceVisible* dans la *DisplayStack*;
3) si l'objet *Visible* destination est actif, redessiner l'entité graphique représentant l'objet sur l'écran dans l'espace de délimitation défini par les attributs *BoxSize* et *Position* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

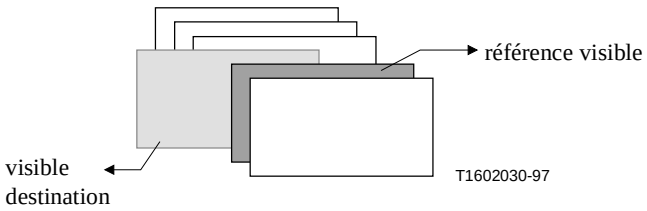


Figure 11/T.172 – Effet de l'action PutBehind

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *ReferenceVisible* sera affecté ou fera référence à un objet *Visible* disponible;
- la pile *DisplayStack* de l'objet *Application* actif contiendra une référence à l'objet *ReferenceVisible*.

Description de la syntaxe:

```
PutBehind            -->    Target,
                            ReferenceVisible
Target               -->    GenericObjectReference
ReferenceVisible     -->    GenericObjectReference
```

SetPaletteRef (*NewPaletteRef*)

Change la couleur de la table de correspondance utilisée pour afficher les couleurs de l'objet *Visible*.

Affecte référence à palette (Nouvelle référence à Palette)

Exécute la séquence d'actions suivante:

1) mettre la palette *PaletteRef* de l'objet *Visible* destination à *NewPaletteRef*;

2) si l'objet *Visible* destination est actif, redessiner l'objet *Visible* en prenant en compte la nouvelle valeur de *PaletteRef*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Visible* disponible;
- *NewPaletteRef* contiendra une référence à un objet *Palette* actif.

Description de la syntaxe:

```
SetPaletteRef        -->  Target,
                          NewPaletteRef
Target               -->  GenericObjectReference
NewPaletteRef        -->  GenericObjectReference
```

31.5 Description formelle

```
Visible Class          -->  Presentable class,
                            OriginalBoxSize,
                            OriginalPosition?,
                            OriginalPaletteRef?
OriginalBoxSize        -->  XLength, YLength
XLength                -->  INTEGER
YLength                -->  INTEGER
OriginalPosition       -->  XYPosition
OriginalPaletteRef     -->  ObjectReference□
```

32 Classe *Bitmap* (Phototrame)

Description: définit le comportement de tableaux de pixels à deux dimensions

classe de base: *Visible*

Sous-classes: aucune

Etat: classe concrète

32.1 Attributs

Le présent sous-paragraphe définit les attributs internes, échangés et hérités de cette classe.

32.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ou son attribut correspondant par défaut dans la classe Application (BitmapContentHook) est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.

32.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Tiling

(Tuilage)

Lorsqu'il est à *True*, cet attribut indique que la phototrame doit être répliquée dans la *BoxSize* disponible de la phototrame.

- Booléen optionnel;
- valeur par défaut: *False*.

OriginalTransparency

(Transparence d'origine)

Cet attribut définit la transparence d'origine des pixels de la phototrame marqués comme étant transparents.

Dans les cas où la représentation codée de la phototrame elle-même spécifie la valeur de l'attribut *Transparency*, la transparence définit par le codage du contenu et celle définie par cet attribut devraient être combinées. L'algorithme exact n'est pas défini dans la présente Recommandation.

- Entier optionnel dans l'intervalle [0, 100] (pourcentage).
- Valeur par défaut: 0%.

32.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Transparency

Transparence

Définit la transparence des pixels de la phototrame marqués comme étant transparents.

- Entier optionnel dans l'intervalle [0, 100] (pourcentage).
- Valeur initiale: valeur de l'attribut *OriginalTransparency*.

32.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

32.3 Comportements internes

Cette classe possède les mêmes comportements que sa classe de base, avec une sémantique identique.

32.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

ScaleBitmap (*XScale, YScale*)

Mise à l'échelle de la phototrame (XScale, YScale)

Si le moteur MHEG-5 implémente l'option *Scaling*, l'effet de cette action est d'ajuster les contenus de la phototrame à la taille (*XScale*, *YScale*). Les moteurs qui n'implémentent pas cette option ignoreront cette action.

Il faut noter que cette action n'affecte pas l'attribut interne *BoxSize* de l'objet *Bitmap*; en d'autres mots, la phototrame est mise à l'échelle, mais sa boîte de délimitation reste la même.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Bitmap* disponible;
- *XScale* et *YScale* seront des entiers positifs.

Définition de la syntaxe:

```
ScaleBitmap          -->  Target,
                          XScale, Yscale
Target               -->  GenericObjectReference
XScale, YScale       -->  GenericInteger
```

SetTransparency (*NewTransparency*)

Affecte la transparence (Nouvelle transparence)

Exécute la séquence d'actions suivante:

1) changer la valeur de l'attribut *Transparency*;
2) si la phototrame est active, la dessiner à nouveau en utilisant la nouvelle valeur de l'attribut *Transparency* selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Bitmap* disponible;
- *NewTransparency* se situe dans l'intervalle [0, 100].

Définition de la syntaxe:

```
ScaleBitmap          -->  Target,
                          NewTransparency
Target               -->  GenericObjectReference
NewTransparency      -->  GenericInteger
```

32.5 Description formelle

```
Bitmap Class           -->  Visible Class,
                            Tiling?,
                            OriginalTransparency?
Tiling                 -->  BOOLEAN
OriginalTransparency   -->  INTEGER
```

33 Classe *LineArt*

Description: définit les fonctionnalités associées à la représentation vectorielle des objets graphiques

classe de base: *Visible*

Sous-classes: *Rectangle*, *DynamicLineArt*

Etat: classe concrète

33.1 Attributs

Cette sous-clause définit les attributs internes, échangés et hérités pour cette classe.

33.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ou son attribut correspondant par défaut (LineArtContentHook) est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.

33.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Bordered BoundingBox

(Bordures de l'espace de délimitation)

L'attribut *BorderedBoundingBox* détermine si l'espace de délimitation, défini par les attributs *BoxSize* et *Position*, est bordé de lignes ou non. Si cet attribut vaut *True*, tous les dessins ultérieurs seront inclus strictement à l'intérieur de la frontière. *OriginalLineWidth*, *OriginalLineStyle* et *OriginalRefLineColour* seront utilisés matérialiser l'espace de délimitation.

- Attribut booléen optionnel.
- Valeur par défaut: *True*.

OriginalLineWidth

(Largeur de tracé à l'origine)

L'attribut *OriginalLineWidth* détermine la largeur de tracé à l'origine de l'objet graphique inclus dans l'espace de délimitation.

L'attribut *OriginalLineWidth* est exprimé en pixels dans l'espace de coordonnées de la scène. Il est spécifié en nombre de pixels en hauteur pour les tracés horizontaux et en nombre de pixels en largeur pour les tracés verticaux.

La résolution d'affichage réelle et la précision du moteur MHEG-5 sont en dehors de la portée de la présente Recommandation.

- Attribut entier optionnel.
- Valeur par défaut: 1.

OriginalLineStyle

(Style de tracé à l'origine)

L'attribut *OriginalLineStyle* détermine le modèle à utiliser pour l'affichage des tracés de l'objet *LineArt*.

- Attribut entier optionnel – 1, 2 ou 3;

  1 signifie plein, 2 signifie tireté, 3 signifie pointillé.
- Valeur par défaut: 1 (plein).

OriginalRefLine Colour

(Couleur de référence de tracé à l'origine)

Couleur de référence initiale pour les tracés utilisés pour afficher l'objet *LineArt*.

La valeur de l'attribut *OriginalRefLineColour* est exprimée soit comme une chaîne représentant une valeur de couleur absolue ou comme un entier représentant un indice à partir de zéro dans une table de correspondance de couleurs. Dans ce dernier cas, l'attribut *OriginalPaletteRef* doit contenir une référence à un objet *Palette* qui est ensuite utilisé pour transformer l'indice en une valeur de couleur réelle.

Le codage des valeurs de couleurs absolues ainsi que la résolution réelle de couleur dans le processus d'affichage sont en dehors de la portée de la présente Recommandation.

- Attribut optionnel.
- Valeur par défaut: "noir".

  L'apparence précise sur l'écran de la valeur par défaut "noir" n'est pas spécifiée totalement dans le présent sous-paragraphe.
- *OctetString* ou Entier.

OriginalRefFill Colour

(Couleur de référence de remplissage à l'origine)

Couleur de référence initiale pour l'intérieur d'un objet *LineArt* fermé.

La valeur de l'attribut *OriginalRefFillColour* est exprimée soit comme une chaîne représentant une valeur de couleur absolue ou comme un entier représentant un indice à partir de zéro dans une table de correspondance de couleurs. Dans ce dernier cas, l'attribut *OriginalPaletteRef* doit contenir une référence à un objet *Palette* qui est ensuite utilisé pour transformer l'indice en une valeur de couleur réelle.

Si l'attribut *OriginalRefFillColour* n'est pas codé, un objet *LineArt* représentant une forme fermée sera affiché avec une couleur de remplissage transparente.

Le codage des valeurs de couleurs absolues ainsi que la résolution réelle de couleur dans le processus d'affichage sont en dehors de la portée de la présente Recommandation.

- Attribut optionnel.
- Valeur par défaut: "transparent".
- *OctetString* ou Entier.

33.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

LineWidth

(Largeur de tracé)

L'attribut *LineWidth* détermine la largeur de tracé des objets graphiques dessinés par les actions graphiques.

- Entier.
- Valeur initiale: *OriginalLineWidth.*

LineStyle

(Style de tracé)

Style de tracé avec lequel l'objet *LineArt* est susceptible d'être tracé.

- 1, 2 ou 3;

  1 signifie plein, 2 signifie tiret, 3 signifie pointillé.
- Valeur initiale: *OriginalLineStyle*.

RefLineColour

(Couleur de référence de tracé)

Couleur de référence pour la couleur de tracé utilisée pour afficher l'objet *LineArt*.

- *OctetString* ou Entier.
- Valeur initiale: *OriginalRefLineColour*.

RefFillColour

Couleur de référence de remplissage

Couleur de référence pour la couleur de fond utilisée pour afficher l'objet *LineArt*.

- OctetString ou Entier.
- Valeur initiale: OriginalRefFillColour.

33.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

33.3 Comportements internes

Cette classe ne définit pas d'attribut interne supplémentaire.

33.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetLineWidth (*NewLineWidth*)

Affecte épaisseur de tracé
(Nouvelle épaisseur de tracé)

Change l'épaisseur des traits d'un objet *LineArt*.

Exécute la séquence d'actions suivante:

1) mettre la valeur de l'attribut *LineWidth* à *NewLineWidth*;
2) si l'objet destination est actif, redessiner immédiatement l'objet destination en prenant en compte la nouvelle valeur de l'attribut *LineWidth* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *LineArt* disponible;
- *NewLineWidth* sera mis ou fera référence à une valeur entière positive.

Description de la syntaxe:

```
SetLineWidth          -->   Target,
                            NewLineWidth
Target                -->   GenericObjectReference
NewLineWidth          -->   GenericInteger
```

SetLineStyle (*NewLineStyle*)

Affecte style de tracé (Nouveau style de tracé)

Change le style de tracé d'un objet *LineArt*.

Exécute la séquence d'actions suivante:

1) mettre la valeur de l'attribut *LineStyle* à *NewLineStyle*;
2) si l'objet destination est actif, redessiner immédiatement l'objet destination en prenant en compte la nouvelle valeur de l'attribut *LineStyle* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Dispositions pour l'utilisation:

- l'objet destination sera un objet *LineArt* disponible;
- *NewLineStyle* sera mis ou fera référence à une valeur entière positive comprise dans l'intervalle [1,3].

Description de la syntaxe:

```
SetLineStyle          -->   Target,
                            NewLineStyle
Target                -->   GenericObjectReference
NewLineStyle          -->   GenericInteger
```

SetLineColour (*NewColour*)

Affecte couleur de tracé (Nouvelle couleur)

Change la couleur de tracé d'un objet *LineArt*.

Exécute la séquence d'actions suivante:

1) mettre la valeur de l'attribut *RefLineColour* à *NewColour*;
2) si l'objet destination est actif, redessine immédiatement l'objet destination en prenant en compte la nouvelle valeur de l'attribut *LineWidth* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

NewColour peut être soit une valeur de couleur absolue ou l'indice à partir de zéro d'une couleur dans la table de correspondance référencée par l'attribut *PaletteRef*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *LineArt* disponible;
- si *RefLineColour* est actuellement mis à une valeur de couleur absolue, *NewColour* sera mis ou référencera une valeur de couleur absolue;
- si *RefLineColour* est actuellement mis à un indice de table de correspondance, *NewColour* sera mis ou référencera un indice de table de correspondance.

Description de la syntaxe:

```
SetLineColour         -->   Target,
                            NewColour
Target                -->   GenericObjectReference
NewColour             -->   NewColourIndex | NewAbsoluteColour
NewColourIndex        -->   GenericInteger
NewAbsoluteColour     -->   GenericOctetString
```

SetFillColour (*NewColour*)

Affecte couleur de remplissage (Nouvelle couleur)

Change la couleur de remplissage d'un objet *LineArt*.

Exécute la séquence d'actions suivante:

1) mettre la valeur de l'attribut *RefFillColour* à *NewColour*;
2) si l'objet destination est actif, redessiner immédiatement l'objet destination en prenant en compte la nouvelle valeur de l'attribut *RefFillColour* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

NewColour peut être soit une valeur de couleur absolue ou l'indice à partir de zéro d'une couleur dans la table de correspondance référencée par l'attribut *PaletteRef*.

Si *NewColour* n'est pas codé, l'attribut *RefFillColour* sera mis à "transparent".

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *LineArt* disponible;
- si *RefFillColour* n'est pas actuellement affecté, *NewColour* sera mis ou fera référence à une valeur de couleur absolue;
- si *RefFillColour* est actuellement mis à une valeur de couleur absolue, *NewColour* sera mis ou référencera une valeur de couleur absolue;
- si *refFillColour* est actuellement mis à un indice de table de correspondance, *NewColour* sera mis ou référencera un indice de table de correspondance.

Description de la syntaxe:

```
SetFillColour          -->   Target,
                             NewColour?
Target                 -->   GenericObjectReference
NewColour              -->   NewColourIndex | NewAbsoluteColour
NewColourIndex         -->   GenericInteger
NewAbsoluteColour      -->   GenericOctetString
```

33.5 Description formelle

```
LineArt Class            -->  Visible Class,
                              BorderedBoundingBox?,
                              OriginalLineWidth?,
                              OriginalLineStyle?,
                              OriginalRefLineColour?,
                              OriginalRefFillColour?
BorderedBoundingBox      -->  BOOLEAN
OriginalLineWidth        -->  INTEGER
OriginalLineStyle        -->  INTEGER
OriginalRefLineColour    -->  Colour
OriginalRefFillColour    -->  Colour
```

34 Classe *Rectangle* (Rectangle)

Description: définit une structure de données concernant les rectangles

classe de base: *LineArt*

Sous-classes: aucune

Etat: classe concrète

34.1 Attributs

Le présent sous-paragraphe définit les attributs internes, échangés et hérités de cette classe.

34.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
BoxSize, Position	*Visible*	En plus du fait de définir l'espace de délimitation de l'objet Rectangle, ces attributs définissent aussi la taille du rectangle lui-même. La taille du rectangle sera telle qu'elle s'ajustera précisément dans l'espace de délimitation. Ceci signifie que les bordures du rectangle occupent LineWidth pixels à l'intérieur de l'espace de délimitation du rectangle.
Bordered BoundingBox	*LineArt*	Cet attribut ne sera pas codé pour cette classe. Il est toujours interprété comme étant à True. Contrairement à sa classe parent, la zone BorderedBoundingBox d'un rectangle (à savoir la forme du rectangle) sera tracée en utilisant LineWidth, LineStyle et RefLineColour, plutôt que OriginalLineWidth, OriginalLineStyle et OriginalRefLineColour.

34.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

34.1.3 Attributs internes propres

Cette classe ne définit aucun attribut interne supplémentaire.

34.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

34.3 Comportements internes

Les comportements internes de cette classe sont les mêmes que ceux de sa classe de base, avec une sémantique identique.

34.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique.

34.5 Description formelle

```
Rectangle Class         -->  LineArt Class
```

35 Classe *DynamicLineArt* (LineArt dynamique)

Description: définit la manière de tracer dynamiquement des objets vectoriels graphiques

classe de base: *LineArt*

Sous-classes: aucune

Etat: classe concrète

35.1 Attributs

Le présent sous-paragraphe définit les attributs internes, échangés et hérités de cette classe.

35.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.

35.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut propre supplémentaire.

35.1.3 Attributs internes propres

Cette classe ne définit aucun attribut interne supplémentaire.

35.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

35.3 Comportements internes

Cette classe ne définit aucun comportement interne supplémentaire.

35.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetBoxSize

(Affecte la taille d'espace)

Cette action redessine l'objet *DynamicLineArt* avec sa nouvelle taille et débarrassé de tous ses tracés précédents, à savoir rempli avec la couleur *OriginalRefFillColour* (dans le cas particulier où la nouvelle taille est exactement la même que la précédente, cette action est identique à une action *Clear*).

Lorsque *BorderedBoundingBox* vaut *True*, la bordure est retracée.

SetPosition, *BringToFront*, *SendToBack*, *PutBefore*, *PutBehind*

(Affecte position, Met au premier plan, Met en arrière-plan, Met devant, Met derrière)

Après chacune de ces actions, l'entité *DynamicLineArt* est enlevée de tous ses dessins précédents, à savoir remplir avec *OriginalRefFillColour*.

Lorsque *BorderedBoundingBox* vaut *True*, la bordure est retracée.

SetLineWidth

(Affecte épaisseur de tracé)

Le fait d'affecter une nouvelle valeur à *LineWidth* ne modifiera pas la largeur de tracé de tous les objets graphiques existants de l'objet *DynamicLineArt*. Elle sera utilisée pour afficher le prochain objet graphique à dessiner.

SetLineStyle

(Affecte style de tracé)

Le fait d'affecter une nouvelle valeur de *LineStyle* ne modifiera pas la largeur de tracé de tous les objets graphiques existants de l'objet *DynamicLineArt* object. Elle sera utilisée pour afficher le prochain objet graphique à dessiner.

SetLineColour

(Affecte couleur de tracé)

Le fait d'affecter une nouvelle valeur de *RefLineColour* ne modifiera pas la largeur de tracé de tous les objets graphiques existants de l'objet *DynamicLineArt* object. Elle sera utilisée pour afficher le prochain objet graphique à dessiner.

SetFillColour

(Affecte couleur de remplissage)

Le fait d'affecter une nouvelle valeur de *RefFillColour* ne modifiera pas la largeur de tracé de tous les objets graphiques existants de l'objet *DynamicLineArt* object. Elle sera utilisée pour afficher le prochain objet graphique à dessiner.

GetLineWidth (*LineWidthVar*)

Récupère l'épaisseur de tracé (Variable à épaisseur de tracé)

Renvoie l'épaisseur de tracé *LineWidth* dans la variable *LineWidthVar*.

Dispositions pour l'utilisation:

- *LineWidthVar* fera référence à un objet *IntegerVariable* actif;
- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
GetLineWidth            -->  Target,
                             LineWidthVar
Target                  -->  GenericObjectReference
LineWidthVar            -->  ObjectReference
```

GetLineStyle (*LineStyleVar*)

Récupère le style de tracé (Variable de style de tracé)

Renvoie le style *LineStyle* courant dans la variable *LineStyleVar*.

Dispositions pour l'utilisation:

- *LineStyleVar* fera référence à un objet *IntegerVariable* actif;
- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
GetLineStyle              -->  Target,
                               LineStyleVar
Target                    -->  GenericObjectReference
LineStyleVar              -->  ObjectReference
```

GetLineColour (*LineColourVar*)

Récupère la couleur de tracé (Variable de couleur de tracé)

Renvoie la valeur courante de *LineColour* dans la variable *LineColourVar*.

Dispositions pour l'utilisation:

- *LineColourVar* fera référence à un objet *OctetStringVariable* actif si la référence *RefLineColour* est spécifiée en tant que *OctetString*. *LineColourVar* fera référence à un objet *IntegerVariable* actif si la référence *RefLineColour* est spécifiée en tant que *Entier*;
- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
GetLineColour             -->  Target,
                               LineColourVar
Target                    -->  GenericObjectReference
LineColourVar             -->  ObjectReference
```

GetFillColour (*FillColourVar*)

Récupère la couleur de remplissage (Variable couleur de remplissage)

Renvoie la référence courante *RefFillColour* dans la variable *FillColourVar*.

Dispositions pour l'utilisation:

- *FillColourVar* fera référence à un objet *OctetStringVariable* actif si la référence *RefFillColour* est spécifiée en tant que *OctetString*. *FillColourVar* fera référence à un objet *IntegerVariable* actif si la référence *RefFillColour* est spécifiée en tant que *Entier*;
- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
GetFillColour             -->  Target,
                               FillColourVar
Target                    -->  GenericObjectReference
FillColourVar             -->  ObjectReference
```

DrawArc (*X, Y, EllipseWidth, EllipseHeight, StartAngle, ArcAngle*)

Tracé d'arc (X,Y, Largeur d'ellipse, Hauteur d'ellipse, Angle de départ, Arc d'angle)

Trace un arc entre *StartAngle* et *StartAngle* + *ArcAngle* (Arc BC dans la Figure 12 ci-dessous).

L'arc est tracé selon la couleur *LineColour*.

Le point *X, Y* est relatif à l'attribut *Position* de l'objet. C'est-à-dire que X=0, Y=0 correspond au coin haut gauche de l'espace de délimitation. Les valeurs de X et Y en dehors de l'espace de délimitation sont autorisées, mais seulement la partie du tracé à l'intérieur de l'espace sera affichée.

Les angles sont exprimés en 64^{e} de degré et sont dans l'intervalle [0, 23039]. *ArcAngle* ne sera jamais égal à 0.

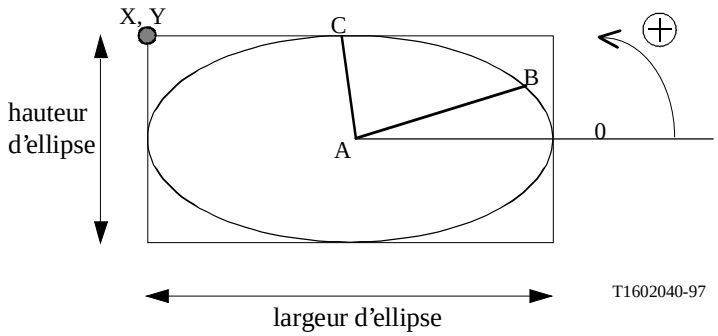


Figure 12/T.172 – Illustration des paramètres de tracé

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawArc                  -->  Target,
                              X,
                              Y,
                              EllipseWidth,
                              EllipseHeight,
                              StartAngle,
                              ArcAngle
Target                   -->  GenericObjectReference
X                        -->  GenericInteger
Y                        -->  GenericInteger
EllipseWidth             -->  GenericInteger
EllipseHeight            -->  GenericInteger
StartAngle               -->  GenericInteger
ArcAngle                 -->  GenericInteger
```

DrawSector (*X, Y, EllipseWidth, EllipseHeight, StartAngle, ArcAngle*)

Tracé de secteur (X,Y, Largeur d'ellipse, Hauteur d'ellipse, Angle de départ, Arc d'angle)

Trace un secteur entre *StartAngle* et *StartAngle* + *ArcAngle* (la surface ABC dans la Figure 12 ci-dessus).

L'arc est tracé selon la couleur *RefLineColour* et la surface est remplie avec la couleur *RefFillColour*.

Le point *X, Y* est relatif à l'attribut *Position* de l'objet. C'est-à-dire que X=0, Y=0 correspond au coin haut gauche de l'espace de délimitation. Les valeurs de X et Y en dehors de l'espace de délimitation sont autorisées, mais seulement la partie du tracé à l'intérieur de l'espace sera affichée.

Les angles sont exprimés en 64^{e} de degré et sont dans l'intervalle [0, 23039]. *ArcAngle* ne sera pas égal à 0.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawSector                --> Target,
                              X,
                              Y,
                              EllipseWidth,
                              EllipseHeight,
                              StartAngle,
                              ArcAngle
Target                    --> GenericObjectReference
X                         --> GenericInteger
Y                         --> GenericInteger
EllipseWidth              --> GenericInteger
EllipseHeight             --> GenericInteger
StartAngle                --> GenericInteger
ArcAngle                  --> GenericInteger
```

DrawLine (*X*1, *Y*1, *X*2, *Y*2)

Tire un trait (*X*1, *Y*1, *X*2, *Y*2)

Tire un trait entre (*X*1, *Y*1) et (*X*2, *Y*2).

Les points *X*1, *Y*1 *et X*2, *Y*2 sont relatifs à l'attribut *Position* de l'objet. Les valeurs en dehors de l'espace de délimitation sont autorisées, mais seule la partie du tracé à l'intérieur de l'espace sera affichée.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawLine                    -->  Target,
                                 X1,
                                 Y1,
                                 X2,
                                 Y2
Target                      -->  GenericObjectReference
X1                          -->  GenericInteger
Y1                          -->  GenericInteger
X2                          -->  GenericInteger
Y2                          -->  GenericInteger
```

DrawOval (*X, Y, EllipseWidth, EllipseHeight*)

Trace une ellipse (X,Y, Largeur d'ellipse, Hauteur d'ellipse)

Trace une ellipse délimitée par le rectangle défini par les paramètres (voir la Figure 12 ci-dessus).

L'ellipse est remplie avec la couleur *RefFillColour*.

Le point *X, Y* est relatif à l'attribut *Position* de l'objet. C'est-à-dire que X=0, Y=0 correspond au coin haut gauche de l'espace de délimitation. Les valeurs de X et Y en dehors de l'espace de délimitation sont autorisées, mais seule la partie du tracé à l'intérieur de l'espace sera affichée.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawOval                    -->  Target,
                                 X,
                                 Y,
                                 EllipseWidth,
                                 EllipseHeight
Target                      -->  GenericObjectReference
X                           -->  GenericInteger
Y                           -->  GenericInteger
EllipseWidth                -->  GenericInteger
EllipseHeight               -->  GenericInteger
```

DrawPolygon (*PointList*)

Trace polygone (Liste de points)

Trace un polygone fermé.

Ce polygone est rempli avec la couleur *RefFillColour*.

La liste *PointList* est une liste de points.

Un point est défini par un couple X et Y relatif à l'attribut *Position* de l'objet. Les valeurs de X et Y en dehors de l'espace de délimitation sont autorisées, mais seule la partie du tracé à l'intérieur de l'espace sera affichée.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawPolygon              -->  Target,
                              PointList
Target                   -->  GenericObjectReference
PointList                -->  Point+
Point                    -->  X,
                              Y
X                        -->  GenericInteger
Y                        -->  GenericInteger
```

DrawPolyline (*PointList*)

Trace polyligne (Liste de points)

Trace une série de traits joints.

La liste *PointList* est une liste de points.

Un point est défini par un couple X et Y relatif à l'attribut *Position* de l'objet. Les valeurs de X et Y en dehors de l'espace de délimitation sont autorisées, mais seule la partie du tracé à l'intérieur de l'espace sera affichée.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawPolyline             -->  Target,
                              PointList
Target                   -->  GenericObjectReference
PointList                -->  Point+
Point                    -->  X,
                              Y
X                        -->  GenericInteger
Y                        -->  GenericInteger
```

DrawRectangle (*X*1, *Y*1, *X*2, *Y*2)
Trace rectangle (*X*1, *Y*1, *X*2, *Y*2)

Trace un rectangle.

Ce rectangle est rempli avec la couleur *RefFillColour*.

Le point haut gauche est *(X*1,*Y*1*)* et le point bas droit est *(X*2 *;Y*2*). (X*1,*Y*1*)* et *(X*2,*Y*2*)* sont relatifs à l'attribut *Position* de l'objet. Les valeurs en dehors de l'espace de délimitation sont autorisées, mais seule la partie du tracé à l'intérieur de l'espace sera affichée.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
DrawRectangle            -->  Target,
                              X1,
                              Y1,
                              X2,
                              Y2
X1                       -->  GenericInteger
Y1                       -->  GenericInteger
X2                       -->  GenericInteger
Y2                       -->  GenericInteger
```

Clear

(Nettoie)

Remplit l'espace de délimitation avec la couleur *OriginalRefFillColour*.

Lorsque *BorderedBoundingBox* vaut *True*, la bordure n'est pas remplie avec *OriginalRefFillColour*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *DynamicLineArt* disponible.

Description de la syntaxe:

```
Clear                    -->  Target
Target                   -->  GenericObjectReference
```

35.5 Description formelle

```
DynamicLineArt Class    -->  LineArt Class
```

36 Classe *Text* (Texte)

Description: définit les attributs et le comportement d'informations textuelles

classe de base: *Visible*

Sous-classes: *EntryField*, *HyperText*

Etat: classe concrète

36.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

36.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ou son attribut correspondant par défaut dans la classe Application (*TextContentHook*) est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe.

36.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

OriginalFont

(Police d'origine)

Indique quelle police utiliser lors de la présentation initiale d'un objet *Text*.

L'attribut *OriginalFont* représente soit un nom de police (qui est résident dans le moteur MHEG-5) ou une référence à un objet *Font*. Dans les deux cas, la police indiquée sera utilisée pour la restitution de l'objet *Text*.

Quand aucune référence à police n'est codée, l'objet *Text* sera présenté en utilisant la police référencée par défaut dans l'objet *Application* actif. Si aucune police n'est référencée là, une police par défaut du moteur MHEG-5 sera utilisée.

NOTE – L'attribut *OriginalFont* fournit une police initiale pour un objet *Text*. De plus, le format de codage du texte peut contenir des séquences d'échappement pour passer d'une police à une autre.

- Attribut optionnel.
- *OctetString* représentant un nom *FontName*, ou référence à un objet *Font*.
- Valeur par défaut: valeur codée par *Application*.

FontAttributes

(Attributs de police)

Cet attribut est utilisé pour positionner des attributs de police spécifiques tels que le style, la taille de caractères, les couleurs de texte et de fond.

Le format de codage exact de l'attribut *FontAttributes* est relatif à la valeur du type de l'objet *Font* mentionné par l'attribut *Font*.

Lorsque aucun *FontAttributes* n'est codé, l'objet *Text* sera présenté en utilisant l'attribut *FontAttributes* par défaut de l'objet *Application* actif. S'il n'y en a pas, aucun attribut spécifique ne sera positionné.

- *OctetString* optionnel.
- Valeur par défaut: valeur codée par *Application*.

TextColour

(Couleur de texte)

Indique la couleur à utiliser pour restituer la couleur d'un objet *Text*. Cet attribut est interprété comme un indice à partir de zéro dans la table de correspondance de couleurs définie par l'attribut *PaletteRef*, ou bien comme une valeur directe, dépendante du type de l'attribut.

- Entier ou *OctetString* optionnel. Un entier sera interprété comme un indice dans une *Palette* ; un *OctetString* comme un valeur de couleur directe.
- Valeur par défaut: valeur codée par *Application*.

BackgroundColour

(Couleur de fond)

Indique la couleur à utiliser pour restituer la couleur de fond d'un objet *Text*. Cet attribut est interprété comme un indice à partir de zéro dans la table de correspondance de couleurs définie par l'attribut *PaletteRef*, ou bien comme une valeur directe, dépendante du type de l'attribut.

- Entier ou *OctetString* optionnel. Un entier sera interprété comme un index dans une *Palette* ; un *OctetString* comme un valeur de couleur directe.
- Valeur par défaut: valeur codée par *Application*.

CharacterSet

(Jeu de caractères)

Identification du jeu de caractères, ou jeu de jeux de caractères à utiliser par défaut pour la restitution du texte. Cet Entier sera codé à l'aide d'une valeur représentant le jeu de caractères. Le domaine applicatif définira un intervalle pour *CharacterSet* ainsi que sa sémantique.

NOTE – L'attribut *CharacterSet* fournit un jeu de caractères initial pour un objet *Text*. De plus, le format de codage du texte peut contenir des séquences d'échappement pour passer d'un jeu de caractères à un autre.

- Entier optionnel.
- Valeur par défaut: la valeur de l'attribut *CharacterSet* de l'objet *Application*, si cet attribut est spécifié.

Horizontal Justification

(Justification horizontale)

La justification *HorizontalJustification* indique comment les lignes de texte sont justifiées relativement aux contours verticaux de la zone de délimitation définie par les attributs *BoxSize* et *Position* de l'objet *Text*.

Cet attribut sera ignoré si la représentation codée de texte elle-même inclut une fonctionnalité similaire spécifiant ce type de restitution. Le domaine applicatif basé sur la présente Recommandation définira chaque *ContentHook* pour lequel cet attribut sera ignoré.

- Attribut optionnel – L'un de ceux-ci : *start* | *end* | *centre* | *justified*.
- Valeur par défaut: *start*.

Vertical Justification

(Justification verticale)

La justification *VerticalJustification* indique comment les lignes de texte sont justifiées relativement aux contours horizontaux de la zone de délimitation définie par les attributs *BoxSize* et *Position* de l'objet *Text*.

Cet attribut sera ignoré si la représentation codée de texte elle-même inclut une fonctionnalité similaire spécifiant ce type de restitution. Le domaine applicatif basé sur la présente Recommandation définira chaque *ContentHook* pour lequel cet attribut sera ignoré.

- Attribut optionnel – L'un de ceux-ci : *start* | *end* | *centre* | *justified*.
- Valeur par défaut: *start*.

LineOrientation

(Orientation de ligne)

L'attribut *LineOrientation* est combiné avec l'attribut *StartCorner* pour déterminer la manière d'organiser les caractères en lignes, et les lignes en séquences.

Cet attribut sera ignoré si la représentation codée de texte elle-même possède une fonctionnalité similaire pour spécifier ce type de restitution. Le domaine applicatif basé sur la présente Recommandation définira chaque *ContentHook* pour lequel cet attribut sera ignoré.

- Attribut optionnel – L'un de ceux-ci : *vertical* | *horizontal*.
- Valeur par défaut: *horizontal*.

NOTE – Pour une orientation de ligne *verticale*, l'orientation de caractère est supposée être normale (à savoir, la ligne entière subit une rotation de 90 degrés par rapport à l'*horizontale*), à moins d'une spécification contraire des contenus de l'objet *Text*.

StartCorner

(Coin début)

L'attribut *StartCorner* contient l'identification d'un coin de présentation où la restitution du texte devrait démarré.

Cet attribut sera ignoré si la représentation codée de texte elle-même inclut une fonctionnalité similaire spécifiant ce type de restitution. Le domaine applicatif basé sur la présente Recommandation définira chaque *ContentHook* pour lequel cet attribut sera ignoré.

- Attribut optionnel – L'un de ceux-ci : *upper-left* | *upper-right* | *lower-left* | *lower-right*.
- Valeur par défaut: *upper-left*.

TextWrapping

(Repli de texte)

Indique si le texte est replié en fin de ligne ou s'il est écrêté.

Si l'attribut *TextWrapping* vaut *True*, le texte est replié, s'il vaut *False*, il est écrêté.

Cet attribut sera ignoré si la représentation codée de texte elle-même inclut une fonctionnalité similaire spécifiant ce type de restitution. Le domaine applicatif basé sur la présente Recommandation définira chaque *ContentHook* pour lequel cet attribut sera ignoré.

- Booléen optionnel;
- valeur par défaut: *False*.

36.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

TextData

(Données de texte)

Valeur de la donnée textuelle de l'objet *Text*.

Si l'attribut *Content* de l'objet *Text* est un contenu *IncludedContent*, *TextData* est mis à l'origine à la valeur de *IncludedContent*.

Si l'attribut *Content* de l'objet *Text* est une référence à une source de donnée externe, *Textdata* contient à l'origine un *OctetString* représentant le contenu de cette source de donnée externe. Dans ce cas, la valeur de l'attribut *ContentHook* pourrait être utilisée pour formater la valeur de *TextData*.

- *OctetString*.
- Valeur initiale : *IncludedContent* ou le contenu de *ReferencedContent*.

Font

(Police)

Police à utiliser lors de la présentation d'un objet *Text*.

L'attribut *Font* représente soit un nom de police (qui est résident dans le moteur MHEG-5) ou une référence à un objet *Font*. Dans les deux cas la police indiquée sera utilisée pour la restitution de l'objet *Text*.

NOTE – L'attribut *Font* fournit la police initiale d'un objet *Font*. De plus, le format de codage du texte peut contenir des séquences d'échappement pour passer d'une police à une autre.

- Attribut optionnel.
- *OctetString* représentant un nom de police *FontName*, ou référence à un objet *Font*.
- Valeur initiale: valeur de l'attribut *OriginalFont*.

36.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

36.3 Comportements internes

Les comportements internes de cette classe ont la même sémantique que ceux de sa classe de base.

36.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData

(Affecte donnée)

Exécute de manière synchrone les actions suivantes:

1) mettre à jour la valeur de l'attribut interne *TextData* de l'objet *Text* destination;

2) exécuter l'action *SetData* comme définie dans la classe *Presentable*.

Les dispositions pour l'utilisation et la description de la syntaxe sont fournies dans la classe *Ingredient*.

GetTextContent (*TextContentVar*)

Récupère contenu de texte (Variable de contenu de texte)

Affecte à la variable référencée par *TextContentVar* la valeur de l'attribut *Content*.

NOTE – Si l'attribut *Content* de l'objet *Text* est inclus, *TextContentVar* renvoie le texte *OctetString*; si l'attribut *Content* de l'objet *Text* est référencé, *TextContentVar* renvoie cette référence à objet.

Dispositions pour l'utilisation:

- *TextContentVar* fera référence à un objet *OctetStringVariable* actif ou à un objet *ContentRefVariable* actif;
- l'objet *Target* sera un objet *Text* disponible.

Description de la syntaxe:

```
GetTextContent        -->  Target,
                           TextContentVar
Target                -->  GenericObjectReference
TextContentVar        -->  ObjectReference
```

GetTextData (*TextDataVar*)

Récupère donnée de texte (Variable de donnée de texte)

Met la variable référencée par *TextDataVar* à la valeur de l'attribut *TextData*.

Dispositions pour l'utilisation:

- *TextDataVar* fera référence à un objet *OctetString* actif;
- l'objet *Target* sera un objet *Text* disponible.

Description de la syntaxe:

```
GetTextData           -->  Target,
                           TextDataVar
Target                -->  GenericObjectReference
TextDataVar           -->  ObjectReference
```

SetFontRef (*NewFont*)

Affecte référence à police (Nouvelle police)

Change la police de caractères utilisée pour restituer un texte.

Exécute la séquence d'actions suivante:

1) mettre la valeur de l'attribut *Font* à *NewFont*;
2) si l'objet *Text* destination est actif, redessiner immédiatement l'objet destination en prenant en compte la nouvelle valeur de *Font* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

NewFont pourrait être une référence à un objet *Font* ou à un nom de police. La présente Recommandation ne définit pas comme les noms de police sont gérés par le moteur MHEG-5.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Text* disponible;
- si *NewFont* référence un objet *Font*, cet objet *Font* sera disponible.

Description de la syntaxe:

```
SetFontRef            -->  Target,
                           NewFont
Target                -->  GenericObjectReference
NewFont               -->  NewFontName | NewFontReference
NewFontName           -->  GenericOctetString
NewFontReference      -->  GenericObjectReference
```

36.5 Description formelle

```
Text Class                -->  Visible Class,
                               OriginalFont?,
                               FontAttributes?,
                               TextColour?,
                               BackgroundColour?,
                               CharacterSet?,
                               HorizontalJustification?,
                               VerticalJustification?,
                               LineOrientation?,
                               StartCorner?,
                               TextWrapping?
OriginalFont              -->  OctetString | ObjectReference
FontAttributes            -->  OctetString
TextColour                -->  Colour
BackgroundColour          -->  Colour
CharacterSet              -->  INTEGER
HorizontalJustification   -->  start | end | centre | justified
VerticalJustification     -->  start | end | centre | justified
LineOrientation           -->  vertical | horizontal
StartCorner               -->  upper-left | upper-right |
                               lower-left | lower-right
TextWrapping              -->  BOOLEAN
```

37 Classe *Stream* (Flux)

Description: définit le comportement d'une composition de média continus (Vidéo, son, RTGraphique) présentés de manière synchronisée

classe de base: *Presentable*

Sous-classes: aucune

Etat: classe concrète

37.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

37.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ou son attribut correspondant par défaut dans la classe Application (StreamContentHook) est obligatoire pour cette classe.
OriginalContent	*Ingredient*	Cet attribut est obligatoire pour cette classe. Il contient une référence à un multiplex complet de médias synchronisés.

37.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Multiplex

(Multiplex)

Liste des composants inclus vidéos, sons et RTGraphiques devant être présentés simultanément. Ils sont appelés *StreamComponents* ci-dessous. Chaque composant de flux possède une balise servant à l'identifier de manière unique dans le flux.

L'attribut *InitiallyActive* de chaque composant *StreamComponent* détermine si le flux élémentaire correspondant est automatiquement joué comme résultat d'un multiplex *Stream* activé pour la première fois.

Il faut noter que tout comportement associé à une synchronisation est toujours accédé via la classe *Stream*. Par exemple, il n'est pas possible d'envoyer l'action à destination d'un objet *Audio*.

- Séquence d'inclusions d'objets *Video*, *Audio* et *RT-Graphics*.

Storage

(Stockage)

Indique si la composition de médias continus est chargée en mémoire avant la restitution ou présentée directement à partir du flux provenant, par exemple, d'un serveur. Pour le moteur MHEG-5, la différence dans la manipulation est que, dans la cas de la *mémoire*, le moteur MHEG-5 synchronisera le flux, tandis que dans le second cas, le *flux* sera synchronisé par le serveur.

- Attribut optionnel – Soit *memory* | *stream*.
- Valeur par défaut: *stream*.

Looping

(Cycle)

Nombre de réalisations de l'objet *Stream*.

Dans le comptage des cycles, le comptage réel a lieu à la fin de l'exécution de l'objet *Stream*. Il en résulte que l'action *SetCounterPosition* sera interprétée dans une boucle donnée.

Lorsqu'un objet *Stream* sera joué et arrêté avant la fin du cycle, l'action "jouer" suivante continuera à partir de ce cycle (et à partir de la position *CounterPosition* à moins d'une spécification contraire par une autre action); en d'autres mots, l'attribut *Looping* représentera le nombre total de cycles.

Lorsqu'un cycle aura atteint la fin de tous les cycles et sera arrêté, une nouvelle activation jouera l'objet *Stream* comme s'il était activé pour la première fois, à savoir en bouclant à nouveau selon l'attribut *Looping*.

- Entier optionnel.
- Valeur par défaut: 1.
- Valeur spéciale: 0 signifiant *infinity*.

37.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

Speed

(Vitesse)

Débit auquel la composition d'un média continu est présentée.

L'attribut *Speed* est un nombre rationnel, représenté par deux entiers *(a, b)*. La vitesse est définie par *a/b*. La sémantique des valeurs suivantes diffère selon que le moteur MHEG-5 a accès ou non à un niveau sous-jacent permettant de supporter le mode "trick".

NOTE – Les modes "trick" peuvent être fournis par le protocole ISO/CEI 13818-6 DSM-CC.

	Pas de mode Trick	**Mode Trick**
–1/1	traité comme 0	retour en arrière
		avance arrière à vitesse normale
0/1	stop	stop
	gèle à la position courante	gèle à la position courante
1/1	avance normale	avance normale
	avance à partir d'un point hors du contrôle du moteur MHEG-5	avance à partir du point où le flux se trouvait lorsqu'il avait quitté son mode précédent

D'autres valeurs de vitesse sont autorisées (par exemple 1/2 or 2/1) dans le mode "trick". Dans l'autre mode, de telles valeurs seront traitées comme un mode normal (1/1) si elles sont positives et comme arrêt (0/1) si elles sont négatives ou nulles.

- Couple d'entiers: un numérateur et un dénominateur optionnel à 1 par défaut.
- Valeur par défaut: 1/1.

CounterPosition

(Position de comptage)

Position temporelle courante du flux à l'intérieur de la durée d'un flux à vitesse normale.

Cet attribut est exprimé en *StreamCounterUnits*.

La définition réelle de *StreamCounterUnit* est en dehors du cadre de la présente Recommandation.

- Entier.
- Valeur par défaut: 0.

CounterEndPosition

(Position en fin de comptage)

Position de la dernière trame d'un flux joué à vitesse normale. Le flux s'arrêtera automatiquement lorsqu'il buttera sur cette position. L'événement *StreamStopped* est généré.

Cet attribut est exprimé en *StreamCounterUnits*.

La définition réelle de *StreamCounterUnit* est en dehors du cadre de la présente Recommandation.

- Entier.
- Valeur par défaut: –1, signifiant *EndOfStream*.

CounterTriggers

(Déclencheurs de compteur)

Liste des valeurs représentant les positions de compteurs du flux à destination desquelles le dérouleur de flux générera des événements *CounterTrigger* concernant ce flux.

Chaque déclencheur possède un identificateur unique à l'intérieur de la liste *CounterTriggers* et une position de compteur exprimée en *StreamCounterUnits*.

- Séquence des structures de données suivantes:
  - identificateur de déclencheur: entier;
  - Position de compteur: entier.
- Valeur initiale: Séquence vide.

37.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. Les événements suivants sont définis en plus:

StreamEvent

(Evénement de flux)

Cet événement est généré automatiquement par le dérouleur de flux lorsque le multiplex *Stream* croise un marqueur spécifique. Le marqueur est reconnu comme une balise pouvant être codée à l'intérieur de la structure de donnée contenant le flux. Il pourrait y avoir plusieurs marqueurs ayant la même identité le long d'un multiplex *Stream*.

- Données associées: *StreamEventTag – OctetString*.

NOTE 1 – Selon la sémantique de la classe *Link*, un objet *Link* destiné à déclencher des *StreamEvents* d'un objet *Stream* spécifique n'ayant pas l'attribut *EventData* affecté, déclenchera tous les *StreamEvents* de cet objet.

NOTE 2 – Le codage des *StreamEvents* peut être effectué en utilisant le protocole DSM-CC ISO/CEI 13818-6.

StreamPlaying

(Déroulement de flux)

Cet événement est généré lorsqu'un multiplex *Stream* commence son déroulement. Plus spécialement, il est généré simultanément avec le premier morceau de donnée *Content* (trame vidéo, échantillon de son) présenté à l'utilisateur.

- Pas de donnée associée.

StreamStopped

(Flux stoppé)

Cet événement est généré lorsqu'un multiplex *Stream* a arrêté son déroulement. Plus spécialement, il est généré dès que le dernier morceau de donnée *Content* (trame vidéo, échantillon de son) a été présenté à l'utilisateur. Il faut noter que l'état *RunningStatus* de l'objet *Stream* n'est pas affecté par l'occurrence d'un événement *StreamStopped*.

- Pas de donnée associée.

CounterTrigger

(Déclencheur de compteur)

Cet événement est généré automatiquement par le dérouleur de flux lorsque *CounterPosition* de l'objet *Stream* traverse une valeur affectée à l'action *SetCounterTrigger*. Il pourrait y avoir plusieurs *CounterTriggers* déclenchés à la même position de compteur d'un objet Stream.

- Donnée associée: *TriggerIdentifier – Integer*.

NOTE – Le codage de la position de compteur à l'intérieur d'un flux peut être effectuée en utilisant le protocole DSM-CC ISO/CEI 13818-6.

37.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet:

Preparation

(Préparation)

Exécute la séquence d'actions suivante:

1) appliquer les trois premières étapes du comportement *Preparation* comme défini dans la classe *Root*;
2) appliquer le comportement *Activation* à tous les *StreamComponents* de l'objet *Stream* ayant l'attribut *InitiallyActive* mis à *True*, dans l'ordre de leur apparition dans le multiplex *Stream*;
3) appliquer les étapes quatre à six du comportement *Preparation* comme défini dans la classe *Root*.

Destruction

(Destruction)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Destruction* à tous les *StreamComponents* du multiplex *Stream*, dans l'ordre inverse de leur apparition dans le multiplex *Stream*;
2) appliquer le comportement *Destruction* comme défini dans la classe de base.

Activation

(Activation)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Activation* comme défini dans la classe de base;
2) commencer à jouer tous les composants *StreamComponents* actifs;
3) mettre l'attribut *RunningStatus* à *True* et génère l'événement *IsRunning*.

L'activation et la désactivation de *StreamComponents* comme *Audio, Video* ou *RTGraphics* correspond à leur mise en marche ou leur arrêt dans un flux. Par exemple, lorsque un flux *Stream* se déroule, le fait d'activer un objet *Audio* rendra ce composant audible (en plus des autres).

Deactivation
(Désactivation)

Exécute la séquence d'actions suivante:

1) si l'attribut *RunningStatus* de l'objet vaut *False*, ignorer ce comportement, sinon:
2) arrêter de dérouler tous les *StreamComponents* actifs;
3) exécuter le comportement *Deactivation* comme défini dans la classe de base.

37.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData
(Affecte donnée)

L'action *SetData* comme définie dans la classe *Ingredient* sera exécutée seulement lorsque le flux est actif.

Clone
(Clone)

Clone ne sera pas à destination d'un objet *Stream*.

SetCounterTrigger (*TriggerIdentifier*, *NewCounterValue*)

Affecte déclencheur de compteur (Identificateur de compteur, Nouvelle valeur de compteur)

Met à jour la liste de déclencheurs *CounterTriggers* d'un objet *Stream*.

Exécute la séquence d'actions suivante:

1) mettre à jour l'ensemble des déclencheurs de la liste *CounterTriggers* de l'objet *Stream* destination, selon les règles suivantes:
   a) si *TriggerIdentifier* est l'identificateur d'un compteur existant dans *CounterTriggers*, le nouveau déclencheur remplace le précédent;
   b) s'il n'y a pas de déclencheur avec *TriggerIdentifier* pour identificateur dans la liste *CounterTriggers*, insérer un nouveau déclencheur d'identificateur *TriggerIdentifier* et de valeur *NewCounterValue* dans la liste *CounterTriggers*;
   c) si *NewCounterValue* n'est pas codée et qu'il existe un déclencheur d'identificateur *TriggerIdentifier* dans la liste *CounterTriggers*, enlever ce déclencheur de la liste *CounterTriggers*;
   d) si *NewCounterValue* n'est pas codé et qu'il n'y a aucun déclencheur d'identificateur *TriggerIdentifier* dans *CounterTriggers*, ne pas tenir compte de cette action;
2) si l'objet *Stream* destination est actif, le moteur MHEG-5 générera des événements *CounterTrigger* selon la nouvelle valeur de la liste *CounterTriggers*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Stream* disponible.

Description de la syntaxe:

```
SetCounterTrigger    -->  Target,
                          TriggerIdentifier,
                          NewCounterValue?
Target               -->  GenericObjectReference
TriggerIdentifier    -->  GenericInteger
NewCounterValue      -->  GenericInteger□
```

SetSpeed (*NewSpeed*)

Affecte vitesse (Nouvelle vitesse)

Change la vitesse de restitution d'un flux.

Exécute la séquence d'actions suivante:

1) mettre la valeur de l'attribut *Speed* de l'objet *Stream* destination à *NewSpeed*;

2) si l'objet *Stream* destination est actif, mettre immédiatement à jour la restitution de l'objet *Stream* en prenant en compte la nouvelle valeur de l'attribut *Speed*.

L'attribut *NewSpeed* est défini comme un rapport Numérateur/Dénominateur.

NOTE 1 – Comme mentionné précédemment, dans un environnement diffusé, les actions jouer et arrêter commenceront dès que possible dans le flux diffusé. Dans d'autres cas, le fait d'affecter une nouvelle vitesse (incluant déroulement normal et arrêt) prendra effet à la position courante du compteur, ou aussi prêt que possible (à savoir à la prochaine I-trame). La position de compteur peut être affectée par l'action appropriée.

NOTE 2 – Si les *modes Trick* ne sont pas supportés par le moteur, la valeur de l'attribut *Speed* peut être mise à n'importe quelle valeur, le moteur les interprétera comme expliqué dans l'attribut interne *Speed*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Stream* disponible.

Description de la syntaxe:

```
SetSpeed             -->  Target,
                          NewSpeed
Target               -->  GenericObjectReference
NewSpeed             -->  Rational
Rational             -->  Numerator,
                          Denominator?
Numerator            -->  GenericInteger□
Denominator          -->  GenericInteger□
```

SetCounterPosition (*NewCounter Position*)

Affecte la position de compteur (Nouvelle position de compteur)

Change la position courante à l'intérieur d'un flux.

Exécute la séquence d'actions suivante:

1) si le moteur MHEG-5 n'est pas fourni avec un niveau de présentation sous-jacent supportant les modes *tricks*, ne pas tenir compte de cette action;
2) mettre la valeur de l'attribut *CounterPosition* de l'objet *Stream* à *NewCounterPosition*;
3) si l'objet *Stream* destination est actif, sauter immédiatement vers la nouvelle position sans changer l'état *RunningStatus* de l'objet *Stream* destination.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Stream* disponible;
- *NewCounterPosition* indiquera une position valide à l'intérieur de l'objet *Stream* destination.

Description de la syntaxe:

```
SetCounterPosition   -->  Target,
                          NewCounterPosition
Target               -->  GenericObjectReference
NewCounterPosition   -->  GenericInteger
```

SetCounterEnd Position (*NewCounterEnd Position*)

Affecte le compteur en position fin (Nouvelle position fin de compteur)

Change la position de fin d'un flux.

Exécute la séquence d'actions suivante:

1) si le moteur MHEG-5 n'est pas fourni avec un niveau de présentation sous-jacent supportant les modes *tricks*, ne pas tenir compte de cette action;
2) mettre la valeur de l'attribut *CounterEndPosition* de l'objet *Stream* à *NewCounterEndPosition*;
3) si l'objet *Stream* destination est actif et que la position *NewCounterEndPosition* a été dépassée, stopper l'objet *Stream* destination.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Stream* disponible;
- *NewCounterPosition* indiquera une position valide à l'intérieur de l'objet *Stream* destination.

Description de la syntaxe:

```
SetCounterPosition   -->  Target,
                         NewCounterPosition
Target               -->  GenericObjectReference
NewCounterPosition   -->  GenericInteger
```

37.5 Description formelle

```
Stream Class         -->  Presentable Class,
                          Multiplex,
                          Storage?,
                          Looping?
Multiplex            -->  StreamComponent+
StreamComponent      -->  Audio class | Video class |
                          RTGraphics class
Storage              -->  memory | stream
Looping              -->  INTEGER
```

38 Classe *Audio* (Audio)

Description: définit les attributs et le comportement d'un flux audio élémentaire dans un multiplex *Stream*. L'objet *Audio* sera un objet *StreamComponent* de l'objet *Stream*

classe de base: *Presentable*

Sous-classes: aucune

Etat: classe concrète

38.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

38.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
RunningStatus	*Root*	Cet attribut exprime qu'un flux est prêt ou désactivé pour être déroulé lorsque l'objet Stream le contenant est actif. L'objet est seulement joué lorsque cet attribut est à True et l'état RunningStatus de l'objet Stream à True.
ContentHook, OriginalContent, Shared	*Ingredient*	Ces attributs ne seront pas codés pour cette classe.

38.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

OriginalVolume

(Volume à l'origine)

Volume de l'objet *Audio* lors de sa première disponibilité.

L'attribut *OriginalVolume* est exprimé en dB (0 dB étant le volume standard pour jouer un son à l'envers). La précision exacte de la restitution du volume est en dehors du champ de la présente Recommandation.

- Entier optionnel.
- Valeur par défaut: 0.

ComponentTag

(Balise de composant)

Un identificateur unique pour le flux audio élémentaire à l'intérieur d'un flux de média multiplexé.

- Entier.

38.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Volume

(Volume)

Volume actuel de l'objet *Audio*, défini de la même manière que *OriginalVolume*.

- Entier.
- Valeur initiale: *OriginalVolume.*

38.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

38.3 Comportements internes

Cette classe ne définit aucun comportement interne supplémentaire.

38.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData
(Affecte donnée)

L'action *SetData* comme définie dans la classe *Ingredient* sera exécutée seulement lorsque le flux *Audio* est inactif.

Clone
Clone

Clone ne sera pas à destination d'un objet *Audio*.

SetVolume
(*NewVolume*)

Affecte volume
(Nouveau volume)

Change le volume d'un objet *Audio*.

Exécute la séquence d'actions suivante:

1) affecter *NewVolume* à l'attribut *Volume* de l'objet *Audio* destination;
2) si le flux audio élémentaire identifié par l'attribut *ComponentTag* est en cours de déroulement, mettre à jour la restitution de l'objet *Audio* en prenant en compte le nouveau *Volume*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Audio* disponible.

Description de la syntaxe:

```
SetVolume            -->   Target,
                           NewVolume
Target               -->   GenericObjectReference
NewVolume            -->   GenericInteger
```

GetVolume (*VolumeVar*)

Récupère le volume (Variable de volume)

Renvoie le volume d'un objet *Audio*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Audio* disponible;
- *VolumeVar* sera un objet *IntegerVariable* actif.

Description de la syntaxe:

```
GetVolume            -->   Target,
                           VolumeVar
Target               -->   GenericObjectReference
VolumeVar            -->   ObjectReference
```

38.5 Description formelle

```
Audio Class            -->  Presentable Class,
                            ComponentTag
                            OriginalVolume?
ComponentTag           -->  INTEGER
OriginalVolume         -->  INTEGER□
```

39 Classe *Video* (Vidéo)

Description: définit les attributs et le comportement d'un flux *Vidéo* élémentaire dans un multiplex *Stream*. L'objet *Video* sera un objet *StreamComponent* de l'objet *Stream*

classe de base: *Visible*

Sous-classes: aucune

Etat: classe concrète

39.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

39.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
RunningStatus	*Root*	Cet attribut exprime qu'un flux est prêt ou désactivé pour être déroulé lorsque l'objet Stream le contenant est actif. L'objet est seulement joué lorsque cet attribut est à True et l'état RunningStatus de l'objet Stream à True.
ContentHook, OriginalContent, Shared	*Ingredient*	Ces attributs ne sont pas codés pour cette classe.
OriginalPaletteRef	*Visible*	Ces attributs ne sont pas codés pour cette classe.

39.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Termination
(Fin)

Cet attribut indique si la dernière trame de la vidéo doit disparaître lorsque la présentation de la vidéo est terminée, ou si elle doit être gelée.

- Attribut optionnel – L'un parmi *freeze* | *disappear*.
- Valeur par défaut: *disappear*.

ComponentTag
(Balise de composant)

Un identificateur unique de flux vidéo élémentaire à l'intérieur d'un flux de média multiplexé.

- Entier.

39.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

39.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

39.3 Comportements internes

Cette classe ne définit aucun comportement interne supplémentaire.

39.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData
(Affecte donnée)

L'action *SetData* comme définie dans la classe *Ingredient* sera exécutée seulement lorsque le flux *Video* est inactif.

Clone
(Clone)

Clone ne sera pas à destination d'un objet *Video*.

ScaleVideo (*XScale*, *YScale*)

Mise à l'échelle de vidéo (XScale, YScale)

Si le moteur MHEG-5 implémente l'option mise à l'échelle, l'effet de cette action sera d'adapter la restitution de la vidéo aux dimensions *XScale* et *YScale*.

Ces paramètres représentent les dimensions finales de la vidéo en nombre de pixels. Ainsi, la représentation graphique de la vidéo peut ne pas conserver son rapport d'aspect d'origine.

Il faut noter que cette action n'affecte pas l'attribut interne *BoxSize* de l'objet *Video*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Video* disponible;
- *XScale* et *YScale* seront des entiers positifs.

Description de la syntaxe:

```
ScaleVideo          -->  Target,
                         XScale, YScale
Target              -->  GenericObjectReference
XScale, YScale      -->  GenericInteger
```

39.5 Description formelle

```
Video Class           -->  Visible Class,
                           ComponentTag
                           Termination?
Termination           -->  freeze | disappear
ComponentTag          -->  INTEGER
```

40 Classe *RTGraghics*(Graphismes Temps Réel)

Description: définit les attributs et le comportement d'objets graphiques non permanents, définis comme flux élémentaire d'un multiplex *Stream*. L'objet *RTGraphics* sera un objet *StreamComponent* de l'objet *Stream*

classe de base: *Visible*

Sous-classes: aucune

Etat: classe concrète

40.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

40.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
RunningStatus	*Root*	Cet attribut exprime qu'un flux est prêt ou désactivé pour être déroulé lorsque l'objet Stream le contenant est actif. L'objet est seulement joué lorsque cet attribut est à True et l'état RunningStatus de l'objet Stream à True.
ContentHook, OriginalContent, Shared	*Ingredient*	Ces attributs ne sont pas codés pour cette classe.

40.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

ComponentTag (Balise de composant)
Un identificateur unique de graphisme non permanent à l'intérieur d'un flux de média multiplexé.
- Entier.

Termination (Fin)
Cet attribut indique si la dernière trame du graphisme doit disparaître lorsque la présentation de la vidéo est terminée, ou si elle doit être gelée.
- Attribut optionnel – L'un parmi *freeze* | *disappear*.
- Valeur par défaut: *disappear*.

40.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

40.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

40.3 Comportements internes

Cette classe ne définit pas d'attribut interne supplémentaire.

40.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetData (Affecte donnée)
L'action *SetData* comme définie dans la classe *Ingredient* sera exécutée seulement lorsque l'objet *RTGraphics* est inactif.

Clone (Clone)
Clone ne sera pas à destination d'un objet *RTGraphics*.

40.5 Description formelle

```
RTGraphics Class        -->  Visible Class,
                             ComponentTag
                             Termination?
ComponentTag            -->  INTEGER
Termination             -->  freeze | disappear
```

41 Classe *Interactible* (*classe Interactivable*)

Description: définit les fonctionnalités associées à un comportement d'interaction de *Visibles*

classe de base: aucune (classe mélangée)

Sous-classes: *Slider*, *HyperText*, *EntryField*, *Button*

Etat: classe abstraite

41.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

41.1.1 Attributs hérités

Cette classe ne possède pas d'attributs hérités.

41.1.2 Attributs propres échangés

Cette classe définit les attributs échangés suivants:

EngineResp

(Responsabilité de moteur)

Détermine où se situe la responsabilité pour la génération du retour visuel à l'utilisateur en tant que résultat de changement de l'attribut interne *HighlightStatus*.

Si *EngineResp* vaut *True*, le moteur MHEG-5 générera l'effet visuel à l'utilisateur lors de tout changement d'état de l'attribut interne *HighlightStatus*. Si *EngineResp* vaut *False*, ne générera pas de retour visuel.

La nature exacte des retours visuels est en dehors du domaine de la présente Recommandation.

- Booléen optionnel.
- Valeur par défaut: *True*.

HighlightRefColour

(Couleur de référence de surbrillance)

Couleur de référence pour les retours visuels générés lorsque l'attribut *HighlightStatus* vaut *True*. La couleur réelle utilisée ne rentre pas dans le champ de la présente Recommandation . Cependant, il est recommandé qu'un moteur MHEG-5 essaie de restituer la surbrillance en utilisant l'attribut *HighlightRefColour*.

Si l'attribut *OriginalPaletteRef* est codé, l'attribut *HighlightRefColour* sera un indice à partir de zéro dans la table de correspondance de couleurs définie par cet objet *Palette*. Sinon, ce sera un *OctetString* contenant le code de la couleur réelle.

- Entier optionnel ou *OctetString*.
- Valeur par défaut: valeur codée par l'application.

41.1.3 Attributs internes propres

Cette classe définit les attributs internes suivants:

HighlightStatus

(Etat surbrillance)

Cet attribut est associé à un certain type de retour visuel à l'utilisateur.

Lorsque *HighlightStatus* et *EngineResp* sont tous les deux à *True*, le moteur MHEG-5 générera un effet visuel à l'utilisateur, par exemple sous la forme d'un trait tiré autour de l'objet *Interactible*. Dans tous les autres cas, aucun retour de ce type ne sera généré. Pour ce qui concerne la génération de ce retour visuel, la couleur *HighlightRefColour* peut être utilisée.

NOTE 1 – Bien que le retour visuel lui-même ne change pas le comportement de cet objet, cet attribut peut être utilisé pour signaler à l'utilisateur que l'objet *Interactible* est prêt pour une interaction, par exemple dans une implémentation de navigation ou l'on va de surbrillance en surbrillance.

NOTE 2 – La seule manière de changer l'attribut interne *HighlightStatus* est d'utiliser l'action *SetHighlightStatus*.

- Valeur booléenne;
- valeur initiale: *False*.

InteractionStatus

(Etat Interaction)

Cet attribut décrit si l'objet *Interactible* est en cours d'interaction ou non.

Si *InteractionStatus* vaut *False*, l'objet *Interactible* n'est pas en cours d'interaction par un utilisateur. La gestion des événements d'entrée d'utilisateur a lieu normalement. Voir le paragraphe 53.

Si *InteractionStatus* vaut *True*, l'objet *Interactible* est en cours d'interaction par un utilisateur. Il en résulte qu'aucun événement de type *UserInput* ne sera généré par l'objet *Scene* actif. Ces événements seront gérés directement par l'objet *Interactible*.

A tout moment, au plus un objet *Interactible* aura son état *InteractionStatus* mis à *True*.

NOTE 1 – La seule manière de changer l'attribut interne *InteractionStatus* est d'utiliser l'action *SetInteractionStatus*.

NOTE 2 – Bien que les liens déclenchés sur des événements *UserInput* ne puissent pas s'amorcer lorsque l'attribut interne *InteractionStatus* vaut *True*, d'autres liens peuvent encore s'amorcer. Cela rend possible, par exemple, l'implémentation de dépassements de temps liés à la phase d'interaction.

- Valeur booléenne;
- valeur initiale: *False*.

41.2 Evénements

Cette classe définit les événements suivants:

Interaction-Completed

(Interaction terminée)

Cet événement est généré en tant que résultat d'un changement d'un objet *Interactible*. L'événement est généré seulement une seule fois par présentation, à savoir généré seulement lorsque l'attribut interne *InteractionStatus* revient à la valeur *False* suite à une interaction.

- Pas de donnée associée.

HighlightOn

(Surbrillance activée)

Cet événement est généré lorsque l'attribut *HighlightStatus* de l'objet *Interactible* passe de la valeur *False* à *True*.

- Pas de donnée associée.

HighlightOff

(Surbrillance désactivée)

Cet événement est généré lorsque l'attribut *HighlightStatus* de l'objet *Interactible* passe de la valeur *True* à *False*.

- Pas de donnée associée.

CursorEnter

(Curseur entre)

Cet événement est généré automatiquement par le moteur MHEG-5 si et seulement si:

1) le moteur implémente l'option curseur "libre déplacement";
2) le curseur en se déplaçant est entré dans la zone définie par l'objet *Interactible*;
3) l'objet *Interactible* est actif.

NOTE – Voir 53.6 sur les mécanismes principaux pour plus de détails sur la génération des événements *CursorLeave* et *CursorEnter*.

CursorLeave

(Curseur quitte)

Cet événement est généré automatiquement par le moteur MHEG-5 si et seulement si:

1) le moteur implémente l'option curseur "libre déplacement";
2) le curseur en se déplaçant a quitté la zone définie par l'objet *Interactible*;
3) l'objet *Interactible* est actif.

NOTE – Voir 53.6 sur les mécanismes principaux pour plus de détails sur la génération des événements *CursorLeave* et *CursorEnter*.

41.3 Comportements internes

Cette classe définit le comportement interne suivant:

Interaction

(Interaction)

Exécute les étapes suivantes de manière synchrone:

1) affecter l'attribut interne *InteractionStatus* à la valeur *True*;
2) générer un retour visuel à l'utilisateur signalant le fait qu'une interaction peut maintenant avoir lieu avec l'objet *Interactible*.

NOTE – La plupart des sous-classes *Interactibles* se déploient sur ce comportement.

41.4 Effet des actions MHEG-5

Cette classe définit les actions MHEG-5 applicables suivantes:

SetInteractionStatus (*NewInteraction-Status*)

Affecte l'état d'interaction (Nouvel état d'interaction)

Cette action est utilisée pour modifier l'attribut interne *InteractionStatus*.

Exécute la séquence d'actions suivante:

1) si *NewInteractionStatus* vaut *True*:
   a) si l'objet *Interactible* ou tout autre *Interactible* de l'objet *Scene* en cours possède son attribut *InteractionStatus* mis à *True*, ne pas tenir compte de cette action. Sinon:
   b) appliquer le comportement *Interaction* de l'objet *Interactible* cible;
2) si NewInteractionStatus vaut False:
   a) si l'objet *Interactible* possède son attribut *InteractionStatus* mis à *False*, ne pas tenir compte de cette action. Sinon:
   b) interrompre immédiatement le comportement *Interaction* de l'objet *Interactible* cible. L'état de l'objet *Interactible* après l'interruption reflétera toute interaction ayant pu avoir lieu avant le moment de l'interruption;
   c) générer l'événement *InteractionCompleted*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Interactible* disponible.

Description de la syntaxe:

```
SetInteractionStatus    -->  Target,
                             NewInteractionStatus
Target                  -->  GenericObjectReference
NewInteractionStatus    -->  GenericBoolean
```

GetInteractionStatus (*InteractionStatus-Var*)

Récupère l'état d'interaction (Variable état d'interaction)

Affecte à la variable référencée par la variable *InteractionStatusVar* la valeur de l'attribut *InteractionStatus*.

Dispositions pour l'utilisation:

- l'objet *Target* sera disponible;
- *InteractionStatusVar* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
GetInteractionStatus      -->   Target,
                                InteractionStatusVar
Target                    -->   GenericObjectReference
InteractionStatusVar      -->   ObjectReference
```

SetHighlightStatus (*NewHighlight Status*)

Affecte l'état de surbrillance (Nouvel état surbrillance)

Change l'état de surbrillance d'un objet *Interactible*.

Exécute de manière synchrone la séquence d'actions suivante:

1) si l'état *HighlightStatus* courant est égal à *NewHighlightStatus*, ne pas tenir compte de cette action. Sinon:
2) affecter l'attribut *HighlightStatus* de l'objet *Interactible* cible à *NewHighlightStatus*;
3) si l'objet *Interactible* cible est actif, si *EngineResp* vaut *True* et si *HighlightStatus* vaut *True*, redessiner l'objet *Interactible* cible pour restituer le retour visuel associé à l'état de surbrillance;
4) si l'objet *Interactible* cible est actif, si *EngineResp* vaut *True* et si *HighlightStatus* vaut *False*, redessiner l'objet *Interactible* cible pour enlever le retour visuel associé à l'état de surbrillance.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Interactible* disponible.

Description de la syntaxe:

```
SetHighlightStatus      -->  Target,
                             NewHighlightStatus
Target                  -->  GenericObjectReference
NewHighlightStatus      -->  GenericBoolean
```

GetHighlightStatus (*HighlightStatusVar*)

Récupère l'état Surbrillance (Variable d'état surbrillance)

Affecte la variable référencée par *HighlightStatusVar* à la valeur de l'attribut *HighlightStatus*.

Dispositions pour l'utilisation:

- l'objet *Target* sera disponible;
- *HighlightStatusVar* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
GetHighlightStatus        -->   Target,
                                HighlightStatusVar
Target                    -->   GenericObjectReference
HighlightStatusVar        -->   ObjectReference
```

41.5 Description formelle

```
Interactible Class      -->  EngineResp?,
                             HighlightRefColour?,
EngineResp              -->  BOOLEAN
HighlightRefColour      -->  Colour
```

42 Classe *Slider* (Curseur)

Description: définit l'entité d'interaction utilisée pour afficher une position à l'intérieur d'un intervalle linéaire

classe de bases: *Visible, Interactible*

Sous-classes: aucune

Etat: classe concrète

42.1 Attributs

Cette clause définit les attributs hérités, échangés et internes de cette classe.

42.1.1 Attributs hérités

Cette classe possède les attributs de sa classe de base avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
BoxSize, Position	*Visible*	En plus de la définition de la zone de délimitation du curseur, ces attributs définissent aussi la taille réelle du curseur. Cette taille sera telle que le curseur remplira complètement sa zone de délimitation.

42.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

Orientation

(Orientation)

Orientation de l'axe principal du curseur.

L'orientation spécifie la direction dans laquelle le curseur se déplace à partir de la valeur minimale vers la valeur maximale.

Bien que la restitution exacte de l'objet *Slider* ne soit pas spécifiée en détail dans la présente Recommandation, celui-ci sera restitué de sorte que son orientation soit conforme à cet attribut.

- Valeur possible: *left* | *right* | *up* | *down*.

InitialValue

(Valeur initiale)

La valeur initiale du curseur, à savoir sa valeur lors de sa première activation faisant suite à la préparation.

La valeur sera un entier et elle sera cohérente avec les valeurs *MinValue*, *MaxValue*, et *StepSize* décrites ci-dessous.

- Entier optionnel.
- Valeur par défaut: *MinValue*.

MinValue

(Valeur minimale)

La plus petite valeur à laquelle l'attribut *SliderValue* peut être affecté.

La valeur sera un entier et elle sera cohérente avec les valeurs *InitialValue*, *MaxValue*, et *StepSize* décrites ci-dessous.

- Entier optionnel.
- Valeur par défaut: 1.

MaxValue

(Valeur maximale)

La plus grande valeur à laquelle l'attribut *SliderValue* peut être affecté.

La valeur sera un entier et elle sera cohérente avec les valeurs *InitialValue*, *MinValue* et *StepSize* décrites ci-dessous.

- Valeur entière.
- Valeur par défaut: aucune.

InitialPortion

(Portion initiale)

Représente une portion de l'intervalle [*MinValue*, *MaxValue*]. Cette valeur sera plus petite ou égale au nombre (*MaxValue – MinValue*).

Cette valeur sera codée si et seulement si la valeur de l'attribut *SliderStyle* est une *valeur proportionnelle dans laquelle la double inégalité suivante est toujours satisfaite*:

$$\text{MinValue} \leq \text{InitialValue} \leq \text{MaxValue} - \text{InitialPortion}$$

- Entier optionnel
- Valeur par défaut: aucune.

StepSize

(Taille de cran)

La plus petite valeur par laquelle la valeur de l'attribut interne *SliderValue* peut être augmentée ou diminuée.

La valeur sera un entier positif cohérent avec *InitialValue*, *MinValue*, et *MaxValue* de la manière suivante:

toutes les valeurs que le curseur peut prendre sont exprimées en:

$$v_i = \text{MinValue} + i \times \text{StepSize},$$

où $i \in \{0 .. (\text{MaxValue} – \text{MinValue})/\text{StepSize}\}$.

Les attributs *InitialValue*, *MinValue*, *MaxValue* et *StepSize* seront conformes aux critères suivants:

$$\text{InitialValue} = \text{MinValue} + M \times \text{StepSize}, \text{ pour } M \in \{0 .. (\text{MaxValue} – \text{MinValue})/\text{StepSize}\}.$$

$$\text{MinValue} < \text{MaxValue}$$

$N \times \text{StepSize} = (\text{MaxValue} – \text{MinValue})$, où N est un entier positif.

- Entier optionnel.
- Valeur par défaut: 1.

SliderStyle

(Style de curseur)

Cet attribut peut prendre les valeurs *normal*, *thermometer*, et *proportional*. L'attribut *SliderStyle* influence la restitution de l'objet *Slider* de la manière suivante:

- si le *SliderStyle* vaut *normal*, l'objet *Slider* est restitué comme un "marqueur" positionné sur un "axe principal" à la position correspondant à l'attribut *SliderValue*;
- si le *SliderStyle* vaut *thermometer*, l'objet *Slider* est restitué comme un "axe principal" rempli depuis le début jusqu'à la position correspondant à l'attribut *SliderValue*;
- si le *SliderStyle* vaut *proportional*, l'objet *Slider* est restitué comme "axe principal" rempli depuis la position correspondant à l'attribut *SliderValue* jusqu'à la position correspondant à la somme des attributs internes *SliderValue* et *Portion*.

La présente Recommandation ne spécifie pas l'aspect précis de cette restitution. Les images suivantes sont fournies à titre d'exemples seulement:

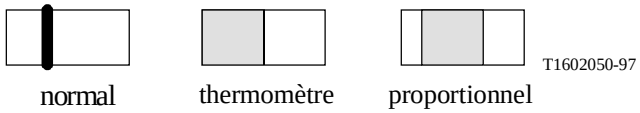


Figure 13/T.172 – Exemples de Curseurs de différents *SliderStyles*

- Attribut optionnel.
- Valeurs possibles: *normal* | *thermometer* | *proportional*.
- Valeur par défaut*: normal* – Spécifie une couleur pouvant être utilisée par le moteur pour restituer l'objet *Slider*.

SliderColour

(Couleur de curseur)

La valeur *SliderColour* est exprimée soit comme une valeur de couleur absolue, ou comme un indice à partir de zéro dans la table de correspondance de couleurs. Dans le dernier cas, l'objet *Slider* doit avoir son attribut *PaletteRef* codé, qui est ensuite utilisé pour traduire l'indice en une couleur réelle.

La présente Recommandation ne spécifie pas le rendu exact de la couleur du curseur. Ceci est fourni en tant que suggestion au moteur MHEG-5 sur un schéma de couleur à utiliser lors d'une restitution de curseur.

La résolution de la couleur réelle lors de la restitution est en dehors du domaine de la présente Recommandation.

- Attribut optionnel.
- Valeur par défaut: valeur codée par l'objet *Application*.

42.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

SliderValue
(Valeur de curseur)

La valeur courante du curseur.

Portion
(Portion)

La valeur courante de l'attribut qui régit la restitution du curseur lorsque son style *SliderStyle* vaut *proportional*.

Si le *SliderStyle* vaut *proportional*, l'objet *Slider* est restitué sous la forme d'un "axe principal" rempli depuis la position correspondant à l'attribut *SliderValue* jusqu'à la position correspondant à la somme des attributs internes *SliderValue* et *Portion*.

42.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

42.3 Comportement interne

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet:

Interaction
(Interaction)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Interaction* comme défini dans la classe *Interactible*;
2) autoriser l'utilisateur à interagir avec l'objet *Slider* en déplaçant le marqueur le long de l'axe principal. La présente Recommandation ne spécifie pas la nature exacte du déplacement. Cependant, le plus petit déplacement de marqueur sera proportionnel à la valeur de l'attribut *StepSize*;
3) lorsque le marqueur est arrêté sur une nouvelle position:
   a) mettre l'attribut *SliderValue* à la valeur correspondant à la nouvelle position de marqueur;
   b) mettre l'attribut *InteractionStatus* à *False*;
   c) générer l'événement *InteractionCompleted*.

42.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

Step
(*NbOfSteps*)
Cran
(nombre de crans)

Affecte une nouvelle valeur à un curseur relativement à sa valeur actuelle.

Exécute la séquence d'actions suivante:

1) si *NbOfSteps* est positif, augmenter la valeur de *SliderValue* de *NbOfSteps* × *StepSize*;
2) si *NbOfSteps* est négatif, diminuer la valeur de *SliderValue* de *NbOfSteps* × *StepSize*;

3) si l'objet *Slider* est actif, redessiner l'objet *Slider* selon la nouvelle valeur de *SliderValue* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif;

4) générer l'événement *InteractionCompleted*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Slider* disponible;
- *NbOfSteps* sera affecté de sorte que:

  $MinValue \leq (NbOfSteps \times StepSize) + SliderValue$,

  et:

  $(NbOfSteps \times StepSize) + SliderValue + Portion \leq MaxValue$

  La valeur de *Portion* dans les expressions ci-dessus sera considérée comme égale à 0 si le style de curseur n'est pas *proportional*.

Description de la syntaxe:

```
Step                --> Target,
                         NbOfSteps
Target              --> GenericObjectReference
NbOfSteps           --> GenericInteger
```

SetSliderValue (*NewSliderValue*)

Affecte la valeur de curseur (Nouvelle valeur de curseur)

Affecte une valeur absolue à un curseur.

Exécute la séquence d'actions suivante:

1) mettre dans l'attribut *SliderValue* de l'objet *Slider* destination la valeur *NewSliderValue*;

2) si l'objet *Slider* est actif, redessiner l'objet *Slider* selon la nouvelle valeur de *SliderValue* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif;

3) générer l'événement *InteractionCompleted*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Slider* disponible.

  NewSliderValue sera comprise dans l'intervalle [*MinValue*, *MaxValue–Portion*].

  La valeur de *Portion* dans les expressions ci-dessus sera prise comme 0 si le style de curseur n'est pas proportionnel;

- (*NewSliderValue – MinValue*) Modulo *StepSize* sera égale à 0.

Description de la syntaxe:

```
SetSliderValue      --> Target,
                        NewSliderValue
Target              --> GenericObjectReference
NewSliderValue      --> GenericInteger
```

GetSliderValue (*SliderValueVar*)

Récupère la valeur de curseur (Variable de valeur de curseur)

Met dans la variable référencée par *SliderValueVar* la valeur de l'attribut *SliderValue*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Slider* disponible;
- *SliderValueVar* fera référence à un objet *IntegerVariable* actif.

Description de la syntaxe:

```
GetSliderValue          -->  Target,
                             SliderValueVar
Target                  -->  GenericObjectReference
SliderValueVar          -->  ObjectReference
```

SetPortion (*NewPortion*)

Affecte portion (Nouvelle portion)

Affecte la taille de la portion représentée par un curseur de style *proportional*.

Exécute la séquence d'actions suivante:

1) mettre l'attribut *Portion* de l'objet *Slider* cible à *NewPortion*;
2) si l'objet *Slider* est actif, redessiner l'objet *Slider* selon la nouvelle valeur de *Portion* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif;
3) générer l'événement *InteractionCompleted*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Slider* disponible de style *proportional*;
- *NewPortion* sera inférieur ou égal à (*MaxValue – SliderValue*).

Description de la syntaxe:

```
SetPortion          -->  Target,
                         NewPortion
Target              -->  GenericObjectReference
NewPortion          -->  GenericInteger
```

GetPortion (*PortionVar*)

Récupère portion (Variable de portion)

Met dans la variable référencée par *PortionVar* la valeur de l'attribut *Portion*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *Slider* disponible de style *proportional*;
- *PortionVar* fera référence à un objet *IntegerVariable* actif.

Description de la syntaxe:

```
GetPortion          -->  Target,
                         PortionVar
Target              -->  GenericObjectReference
PortionVar          -->  ObjectReference
```

42.5 Description formelle

```
Slider Class          --> Visible Class,
                          Interactible Class,
                          Orientation,
                          Max Value,
                          Min Value?,
                          InitialValue?,
                          InitialPortion?,
                          StepSize?,
                          SliderStyle?,
                          SliderColour?□
Orientation           --> left | right | up | down
InitialValue          --> INTEGER
InitialPortion        --> INTEGER
Min Value             --> INTEGER
Max Value             --> INTEGER
StepSize              --> INTEGER
SliderStyle           --> normal | thermometer | proportional
SliderColour          --> Colour□
```

43 Classe *EntryField* (Champ d'entrée)

Description: définit une entité d'interaction utilisée par l'utilisateur final pour éditer ou modifier un texte

classe de bases: *Text, Interactible*

Sous-classes: aucune

Etat: classe concrète

43.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

43.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec la même sémantique.

43.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

InputType (Type en entrée) — Type de caractères autorisés. Si cet attribut vaut *alpha*, seuls les caractères n'étant pas des chiffres (i.e. 0123456789) seront acceptés. Si cet attribut vaut *numeric*, seuls les chiffres seront acceptés. Si cet attribut vaut *any*, tous les caractères seront acceptés. Si cet attribut vaut *listed*, seuls les caractères fournis par l'attribut *CharList* de *EntryField* seront acceptés. Ceci fournit la possibilité de personnaliser un filtre d'entrée.

NOTE – la valeur *any* autorise aussi les symboles en entrée (e.g. &*$#@!).

- Valeurs possibles: *alpha* | *numeric* | *any* | *listed*.
- Valeur par défaut: *any*.

CharList

(Liste de caractères)

Caractères pouvant être saisi dans le champ *EntryField*.

Cet attribut sera toujours codé lorsque *InputType* vaut *listed*. Sinon, il ne le sera pas.

- *OctetString* optionnel.
- Valeur par défaut: aucune.

ObscuredInput

(Entrée en mode masqué)

Indique d'afficher en écho les caractères entrés. Cette fonction peut servir à la saisie des mots de passe.

Si cet attribut vaut *True*, l'écho renvoyé sur les caractères entrés ne sera pas une forme lisible. Sinon, les caractères rentrés seront renvoyés en écho sous forme visible.

- Valeur booléenne optionnelle.
- Valeur par défaut: *False*.

MaxLength

(Longueur maximale)

Fournit le nombre maximal de caractères attendus. Lorsque ce nombre maximal de caractères est atteint dans le champ *EntryField*, l'événement *EntryFieldFull* est généré.

Si *MaxLength* vaut 0, le nombre de caractères attendus est indéfini. Dans ce cas le moteur MHEG-5 fournira à l'utilisateur des moyens pour terminer son interaction.

- Entier optionnel.
- Valeur par défaut: 0.

43.1.3 Attributs internes propres

Cette classe définit les attributs internes supplémentaires suivants:

EntryPoint

(Point d'entrée)

Définit (sous forme d'indice à partir de zéro) où sera placé le caractère suivant à entrer dans le champ *EntryField*:

- si cet attribut vaut 0, le caractère suivant sera placé avant le premier caractère de *EntryField*;
- si cet attribut est supérieur ou égal à la longueur du texte actuellement dans *EntryField*, le caractère suivant sera accolé à la fin de *EntryField*;
- pour les autres valeurs *n* de cet attribut, la caractère suivant sera inséré après le n^e caractère de *EntryField*.

- Valeur entière.
- Valeur initiale: 0.

OverwriteMode

(Mode remplacement)

Détermine si les nouveaux caractères entrés remplacent les caractères du texte existant ou sont insérés parmi eux.

Si cet attribut vaut *True*, chaque caractère entré dans *EntryField* remplace le caractère placé au *point d'entrée*, tous les autres caractères ne sont pas affectés.

Si cet attribut vaut *False*, l'entrée est insérée juste avant le caractère placé au *point d'entrée*.

Si le point *EntryPoint* vaut une valeur supérieure ou égale à la longueur du texte de *EntryField*, les caractères entrés sont accolés à la fin de *EntryField* sans tenir compte de la valeur de cet attribut.

- Booléen.
- Valeur par défaut: *False*.

43.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, les événements suivants sont définis:

Interaction-Completed

(Interaction terminée)

Cet événement est généré comme résultat de la fin d'un processus de modification d'un champ *EntryField*.

EntryFieldFull

(Champ d'entrée plein)

Cet événement est généré lorsque la capacité maximale de *EntryField* est atteinte. Cet événement sera généré seulement lorsque l'attribut *MaxLength* est codé, et dans ce cas lorsque le nombre de caractères de *EntryField* atteint la valeur *MaxLength*.

L'événement est asynchrone.

L'action *SetData* ne générera pas d'événement *EntryFieldFull*.

- Pas de donnée associée.

43.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet:

Interaction

(Interaction)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Interaction* comme défini dans la classe *Interactible*;
2) autoriser l'utilisateur à modifier l'attribut *TextData* de l'objet *EntryField* en entrant des caractères;

3) à la suite de chaque entrée de caractère, mettre à jour l'attribut *TextData* en conséquence (en prenant en compte la valeur des attributs *EntryPoint* et *Overwrite*), mettre à jour l'attribut *EntryPoint*;

4) lorsque la saisie est terminée (sur décision de l'utilisateur ou à l'initiative de l'application terminant l'action *SetInteractionStatus*):

   a) mettre l'attribut *InteractionStatus* à la valeur False;

   b) générer l'événement *InteractionCompleted*.

43.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetOverwriteMode (*NewOverwriteMode*)

Fait passer un champ de saisie du mode ajout au mode remplacement et vice versa.

Affecte le mode remplacement (Nouveau mode remplacement)

Exécute la séquence d'actions suivante:

1) si le mode actuel *OverwriteMode* est égal à *NewOverwriteMode*, ne pas tenir compte de cette action. Sinon:

2) affecter *NewOverwriteMode* à l'attribut *OverwriteMode* du champ *EntryField*;

3) si le champ *EntryField* cible est actif, mettre à jour la représentation visible du champ *EntryField* selon la nouvelle valeur de *OverwriteMode*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *EntryField* disponible.

Description de la syntaxe:

```
SetOverwriteMode     -->  Target,
                          NewOverwriteMode
Target               -->  GenericObjectReference
NewOverwriteMode     -->  GenericBoolean
```

GetOverwriteMode (*OverwriteModeVar*)

Affecte la valeur de l'attribut *OverwriteMode* à la variable référencée par *OverwriteModeVar*.

Récupère le mode remplacement (Variable mode remplacement)

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *EntryField* disponible;
- *OverwriteModeVar* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
GetOverwriteMode     -->  Target,
                          OverwriteModeVar
Target               -->  GenericObjectReference
OverwriteModeVar     -->  ObjectReference
```

SetEntryPoint (*NewEntryPoint*)

Affecte le point d'entrée (Nouveau point d'entrée)

Change la position du point d'entrée d'un champ de saisie.

Exécute la séquence d'actions suivante:

1) affecter *NewEntryPoint* dans *EntryPoint* de l'entrée *EntryField* destination;

2) si le champ *EntryField* est actif, met à jour la représentation visible de *EntryField* selon la nouvelle valeur de *EntryPoint*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *EntryField*;
- *NewEntryPoint* sera supérieur ou égal à 0.

Description de la syntaxe:

```
SetEntryPoint        -->  Target,
                          NewEntryPoint
Target               -->  GenericObjectReference
NewEntryPoint        -->  GenericInteger
```

GetEntryPoint (*EntryPointVar*)

Récupère le point d'entrée (Variable de point d'entrée)

Met la variable référencée par *EntryPointVar* la valeur de l'attribut *EntryPoint*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *EntryField* disponible;
- *EntryPointVar* fera référence à un objet *IntegerVariable* actif.

Description de la syntaxe:

```
GetEntryPoint        -->  Target,
                          EntryPointVar
Target               -->  GenericObjectReference
EntryPointVar        -->  ObjectReference
```

43.5 Description formelle

```
EntryField Class     -->  Text Class,
                          Interactible Class,
                          InputType?,
                          CharList?,
                          ObscuredInput?,
                          MaxLength?
InputType            -->  alpha | numeric | any | listed
CharList             -->  OctetString
ObscuredInput        -->  BOOLEAN
MaxLength            -->  INTEGER
```

44 Classe *HyperText* (Hypertexte)

Description:	la classe *HyperText* est une sous-classe de la classe *Text*, avec la propriété spécifique d'autoriser l'association de portions de textes avec des informations. Ces portions de texte sont appelées des ancres (*anchors*) dans la suite du présent paragraphe
classe de bases:	*Text, Interactible*
Sous-classes:	aucune
Etat:	classe concrète

44.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

44.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base avec la même sémantique.

44.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

44.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

LastAnchorFired	Balise de la dernière ancre amorcée.
(Dernière ancre amorcée)	• *OctetString*. • Valeur initiale : chaîne vide.

44.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique. De plus, l'événement suivant est défini:

AnchorFired	Signale que l'utilisateur a sélectionné l'une des ancres de l'objet *Hypertext*.
(Ancre amorcée)	• Donnée associée: la balise de l'ancre amorcée – *OctetString*.

44.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet:

Interaction	Exécute la séquence d'actions suivante:
(Interaction)	1) appliquer le comportement *Interaction* comme hérité de la classe *Interactible*; 2) autoriser l'utilisateur à déplacer l'état affiché à travers l'ensemble des ancres de l'objet *Hypertext* et à sélectionner l'ancre affichée. A chaque sélection d'ancre, un événement *AnchorFired* est généré;

3) lorsque l'interaction est finie (soit à l'initiative de l'utilisateur sur fin de l'interaction ou à l'initiative de l'application activant l'action *SetInteractionStatus*):

a) mettre l'attribut *InteractionStatus* à *False*;

b) générer l'événement *InteractionCompleted*.

44.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, l'action MHEG-5 applicable suivante est définie:

GetLastAnchor Fired (*LastAnchor FiredVar*)

Récupère la dernière ancre amorcée (Variable de dernière ancre amorcée)

Met la valeur de l'attribut *LastAnchorFired* dans la variable référencée par *LastAnchorFiredVar*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *HyperText* disponible;
- *LastAnchorFired* fera référence à un objet *OctetStringVariable* disponible.

Description de la syntaxe:

```
GetLastAnchorFired   -->  Target,
                          LastAnchorFiredVar
Target               -->  GenericObjectReference
LastAnchorFiredVar   -->  ObjectReference
```

44.5 Description formelle

```
HyperText Class        -->  Text Class,
                            Interactible Class
```

45 Classe *Button* (Bouton)

Description: définit les fonctionnalités associées à la restitution et l'interaction avec des boutons à un ou deux états

classe de base: *Visible*, *Interactible*

Sous-classes: *HotSpot*, *PushButton*

Etat: classe abstraite

45.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

45.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec les contraintes suivantes:

Nom d'attribut	Défini dans	Contraintes et besoins
ContentHook?	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
OriginalContent	*Ingredient*	Cet attribut ne sera pas codé pour cette classe.
InteractionStatus	*Interactible*	Cet attribut n'est pas défini pour la classe Button.
BoxSize, Position	*Visible*	En plus de la définition de la zone de délimitation du bouton, ces attributs définissent aussi la taille réelle du bouton. Cette taille sera telle que le bouton remplira complètement sa zone de délimitation.

45.1.2 Attributs propres échangés

Cette classe définit l'attribut échangé supplémentaire suivant:

ButtonColour

(Couleur de bouton)

Spécifie une couleur pouvant être utilisée par le moteur MHEG-5 pour restituer l'objet *Button*.

La valeur de l'attribut *ButtonColour* est exprimée soit comme une valeur de couleur absolue ou comme un indice à partir de zéro dans une table de correspondance de couleurs. Dans le dernier cas, l'objet *Button* doit avoir l'attribut *PaletteRef* codé, qui est ensuite utilisé pour traduire l'indice dans une valeur de couleur réelle.

La restitution exacte de la couleur utilisée pour le bouton n'est pas spécifiée dans la présente Recommandation. Ceci est fournit en tant que suggestion au moteur MHEG-5 d'un schéma de couleur à utiliser pour la restitution de bouton.

La résolution réelle de couleur dans le processus de restitution ne rentre pas dans le champ de la présente Recommandation.

- Attribut optionnel.
- Valeur par défaut: valeur codée par l'*Application*.

45.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

SelectionStatus

(Etat sélection)

Chaque bouton peut stocker un bit d'information. Cet attribut est à *True* lorsque l'objet *Button* est dans l'état sélectionné. L'état sélectionné est entré comme le résultat de l'invocation de l'action *Select*.

- Valeur booléenne.
- Valeur initiale: *False*

45.2 Evénements

IsSelected

(Est sélectionné)

Cet événement est généré lorsque l'état *SelectionStatus* du bouton passe de *False* à *True*.

- Aucune donnée associée.

IsDeselected

(N'est plus sélectionné)

Cet événement est généré lorsque l'état *SelectionStatus* du bouton passe de *True* à *False*.

- Aucune donnée associée.

45.3 Comportements internes

La sémantique des comportements internes suivants a changé par rapport à ceux de la classe de base de cet objet:

Interaction

(Interaction)

Ce comportement n'est pas défini pour la classe *Button*.

NOTE – Il en résulte que la classe *Button* ne générera aucun événement *InteractionCompleted*.

Selection

(Sélection)

Exécute la séquence d'actions suivante:

1) mettre l'état *SelectionStatus* à *True*;
2) si l'attribut *EngineResp* est à *True* et l'objet *Button* est actif, redessiner l'objet *Button* en prenant en compte la nouvelle valeur de l'état *SelectionStatus* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif;
3) générer l'événement *IsSelected*.

Deselection

(Désélection)

Exécute la séquence d'actions suivante:

1) mettre l'état *SelectionStatus* à *False*;
2) si l'attribut *EngineResp* est à *True* et l'objet *Button* est actif, redessiner l'objet *Button* en prenant en compte la nouvelle valeur de l'état *SelectionStatus* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif;
3) générer l'événement *IsDeselected*.

45.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

SetInteractionStatus

(Affecte l'état interaction)

Cette action ne sera pas adressée à un objet *Button*.

GetInteraction-Status

(Récupère l'état interaction)

Cette action ne sera pas adressée à un objet *Button*.

Select

(Sélectionne)

Exécute la séquence d'actions suivante:

1) si l'état *SelectionStatus* vaut actuellement *True*, ne pas tenir compte de cette action;

2) si l'état *SelectionStatus* vaut actuellement *False*, appliquer le comportement *Selection* de l'objet cible *Button*.

Disposition pour l'utilisation:

- L'objet *Target* sera un objet *Button* disponible.

Description de la syntaxe:

```
Select              -->  Target

Target              -->  GenericObjectReference
```

Deselect

(Désélectionne)

Exécute la séquence d'actions suivante:

1) si l'état *SelectionStatus* vaut actuellement *False*, ne pas tenir compte de cette action;

2) si l'état *SelectionStatus* vaut actuellement *True*, appliquer le comportement *Deselection* de l'objet cible *Button*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *Button* disponible.

Description de la syntaxe:

```
Deselect            -->  Target

Target              -->  GenericObjectReference
```

45.5 Description formelle

```
Button Class          -->   Visible Class,
                            Interactible Class,
                            ButtonColour?

ButtonColour          -->   Colour
```

46 Classe *Hotspot* (Zone cliquable)

Description: définit des zones rectangulaires non étiquetées invisibles sur l'écran pouvant réagir aux interactions de l'utilisateur pour produire des événements *IsSelected*

classe de base: *Button*

Sous-classes: aucune

Etat: classe concrète

46.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

46.1.1 Attributs hérités

Cette classe possède tous les attributs de sa classe de base, avec une sémantique identique, sauf pour l'attribut suivant:

SelectionStatus

(Etat sélection)

La restitution d'un objet *Hotspot* dépendra de l'attribut *SelectionStatus*. Lorsque les attributs *SelectionStatus* et *EngineResp* valent tous les deux *True*, l'objet *Hotspot* sera restitué de manière à signaler à l'utilisateur qu'une sélection a eu lieu. L'attribut *ButtonColour* peut être utilisé pour ce processus de restitution. Dans tous les autres cas, l'objet *HotSpot* n'aura aucun effet visuel sauf si une telle restitution est prévue par l'état *HighlightStatus* de sa classe de base.

46.1.2 Attributs propres échangés

Cette classe ne définit aucun attribut échangé supplémentaire.

46.1.3 Attributs internes propres

Cette classe ne définit pas d'attribut interne supplémentaire.

46.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

46.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

46.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. La sémantique de l'action MHEG-5 suivante a changé par rapport à celle de la classe de base:

Select

(Sélectionne)

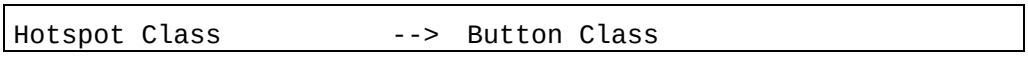
Exécute la séquence d'actions suivante:

1) appliquer le comportement *Selection* comme défini dans la classe de base;

2) appliquer le comportement *Deselection* comme défini dans la classe de base.

Les dispositions pour l'utilisation et la description de la syntaxe de l'action restent inchangées.

46.5 Description formelle



47 Classe *PushButton* (Bouton poussoir)

Description:	définit des zones rectangulaires étiquetées sur l'écran qui peuvent réagir aux interactions de l'utilisateur pour produire des événements *IsSelected*
classe de base:	*Button*
Sous-classes:	*SwitchButton*
Etat:	classe concrète

47.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

47.1.1 Attributs hérités

Cette classe possède les mêmes attributs que sa classe de base, avec une sémantique identique, sauf pour l'attribut suivant:

SelectionStatus

(Etat sélection)

La restitution d'un objet *PushButton* dépendra de l'attribut *SelectionStatus*. Lorsque les attributs *SelectionStatus* et *EngineResp* sont tous les deux à *True*, L'objet *PushButton* sera restitué de manière à signaler à l'utilisateur qu'une sélection a eu lieu. Cette restitution rendra compte du fait qu'un bouton a été pressé. Dans les autres cas, l'objet *Button* sera restitué de manière à signaler à l'utilisateur qu'aucune sélection n'a eu lieu. Cette restitution rendra compte du fait qu'aucun bouton n'a été pressé.

47.1.2 Attributs propres échangés

Cette classe définit les attributs échangés supplémentaires suivants:

OriginalLabel

(Etiquette d'origine)

Bout de texte représentant l'étiquette initiale du bouton poussoir.

- *OctetString* optionnel.
- Valeur par défaut: chaîne vide.

CharacterSet

(Jeu de caractères)

Identification de jeu de caractères ou d'ensemble de jeux de caractères à utiliser par défaut pour restituer l'étiquette. Cet entier sera codé avec une valeur représentant le jeu de caractères. Le domaine applicatif définira un intervalle pour *CharacterSet* ainsi que sa sémantique.

NOTE – L'attribut *CharacterSet* fournit le jeu de caractères initial d'une étiquette. De plus, le format de codage du texte peut contenir des séquences d'échappement pour passer d'un jeu à l'autre.

- Entier optionnel.
- Valeur par défaut: la valeur de l'attribut *CharacterSet* de l'objet *Application*, si cet attribut est spécifié.

47.1.3 Attributs internes propres

Cette classe définit l'attribut interne supplémentaire suivant:

Label

(Etiquette)

Etiquette du bouton poussoir.

- *OctetString* optionnel.
- Valeur initiale: valeur de l'attribut *OriginalLabel*.

47.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

47.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

47.4 Effet des actions MHEG-5

Cette classe possède le même ensemble d'actions MHEG-5 que sa classe de base, avec les changements et extensions suivantes:

Select

(Sélectionne)

Exécute la séquence d'actions suivante:

1) appliquer le comportement *Selection* comme défini dans la classe de base;

2) appliquer le comportement *Deselection* comme défini dans la classe de base.

Les dispositions pour l'utilisation et la description de la syntaxe de l'action restent inchangées.

SetLabel (*NewLabel*)

Affecte étiquette (Nouvelle étiquette)

Change l'étiquette d'un objet *PushButton*.

Exécute la séquence d'actions suivante:

1) mettre l'attribut *Label* de l'objet *PushButton* cible à *NewLabel*;

2) si l'objet *PushButton* est actif, redessiner l'objet *PushButton* en prenant en compte la nouvelle valeur de *Label* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *PushButton* disponible.

Définition de la syntaxe:

```
SetLabel            -->  Target,
                         NewLabel
Target              -->  GenericObjectReference
NewLabel            -->  GenericOctetString
```

GetLabel (*LabelVar*)

Récupère étiquette (Variable d'étiquette)

Met la variable référencée par *LabelVar* à la valeur de l'attribut *Label*.

Disposition pour l'utilisation:

- *LabelVar* fera référence à un objet *OctetStringVariable* actif.

Description de la syntaxe:

```
GetLabel                  -->  Target,
                               LabelVar
Target                    -->  GenericObjectReference
LabelVar                  -->  ObjectReference
```

47.5 Description formelle

```
PushButton Class          -->  Button Class,
                               OriginalLabel?,
                               CharacterSet?
OriginalLabel             -->  □□□□□□□□□□□□
```

48 Classe *SwitchButton* (Case à cocher, Bouton radio, Bouton poussoir)

Description: définit une structure de données pour traiter les zones rectangulaires étiquetées sur l'écran qui peuvent réagir aux interactions de l'utilisateur pour produire des événements *IsSelected* et *IsDeselected*

classe de base: *PushButton*

Sous-classes: aucune

Etat: classe concrète

48.1 Attributs

Cette classe possède les mêmes attributs que sa classe de base, avec une sémantique identique, sauf pour l'attribut suivant:

SelectionStatus (Etat sélection)

La restitution d'un objet *SwitchButton* dépendra de l'attribut *SelectionStatus*.

Lorsque les attributs *SelectionStatus* et *EngineResp* valent tous les deux *True*, l'objet *SwitchButton* sera restitué de manière à signaler à l'utilisateur qu'une sélection a eu lieu. Cette restitution rendra compte du fait qu'un bouton radio ou une case à cocher a été sélectionné ou qu'un bouton poussoir a été poussé. Ceci dépend de l'attribut *ButtonStyle*. Dans les autres cas, l'objet *Button* sera restitué de manière à signaler à l'utilisateur qu'aucune sélection n'a eu lieu. Cette restitution rendra compte du fait qu'aucun bouton radio ou case à cocher n'a été sélectionné, ou qu'aucun bouton n'a été pressé.

48.1.1 Attributs hérités

Cette classe possède les mêmes attributs que sa classe de base, avec une sémantique identique.

48.1.2 Attributs propres échangés

Cette classe définit l'attribut échangé supplémentaire suivant:

ButtonStyle (Style de bouton)

Style de présentation de l'objet *SwitchButton*.

- valeurs possibles: pushbutton | radiobutton | checkbox.

48.1.3 Attributs internes propres

Cette classe ne définit aucun attribut interne supplémentaire.

48.2 Evénements

Cette classe possède les mêmes événements que sa classe de base, avec une sémantique identique.

48.3 Comportements internes

Cette classe possède les mêmes comportements internes que sa classe de base, avec une sémantique identique.

48.4 Effet des actions MHEG-5

Cette classe possède le même jeu d'actions MHEG-5 que sa classe de base, avec une sémantique identique. De plus, les actions MHEG-5 applicables suivantes sont définies:

GetSelectionStatus (*SelectionStatus-Var*)

Récupère l'état sélection (Variable état de sélection)

Renvoie la valeur de l'attribut *SelectionStatus* sous la forme d'un Booléen dans la variable référencée par le paramètre *SelectionStatusVar*.

Dispositions pour l'utilisation:

- l'objet *Target* sera un objet *SwitchButton* disponible;
- *SelectionStatusVar* fera référence à un objet *BooleanVariable* actif.

Description de la syntaxe:

```
GetSelectionStatus   -->  Target,
                          SelectionStatusVar
Target               -->  GenericObjectReference
SelectionStatusVar   -->  ObjectReference
```

Select

(Sélectionne)

Si l'attribut *SelectionStatus* est à *True*, ne pas tenir compte de cette action. Sinon, invoque le comportement *Selection*.

Deselect

(Désélectionne)

Si l'attribut *SelectionStatus* est à *False*, ne pas tenir compte de cette action. Sinon, invoque le comportement *Deselection*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *SwitchButton* disponible.

Description de la syntaxe:

```
Deselect             -->   Target
Target               -->   GenericObjectReference
```

Toggle

(Bascule)

Si l'attribut *SelectionStatus* est à *False*, invoque le comportement *Selection*. Sinon, invoque le comportement *Deselection*.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *SwitchButton* disponible.

Description de la syntaxe:

```
Toggle              -->   Target

Target              -->   GenericObjectReference
```

SetLabel

(Affecte étiquette)

Exécute la séquence d'actions suivante:

1) mettre l'attribut *Label* à la nouvelle valeur;
2) si l'objet *SwitchButton* est actif, redessiner l'objet *SwitchButton* en prenant en compte la nouvelle valeur de *Label* et selon sa position dans la pile *DisplayStack* de l'objet *Application* actif.

Disposition pour l'utilisation:

- l'objet *Target* sera un objet *SwitchButton* disponible.

La syntaxe de cette action est définie dans la classe *PushButton*.

48.5 Description formelle

```
SwitchButton Class      -->  PushButton Class,
                             ButtonStyle
ButtonStyle             -->  pushbutton | radiobutton |
                             checkbox
```

49 Classe *Action* (Action)

Description: la classe *Action* est une liste d'actions élémentaires à exécuter de manière synchrone

la classe *Action* n'hérite d'aucune autre classe MHEG-5 ; il en résulte que les objets *Action* ne peuvent pas être référencés individuellement

classe de base: aucune

Sous-classes: aucune

Etat: classe concrète

49.1 Attributs

Le présent sous-paragraphe définit les attributs hérités, échangés et internes de cette classe.

49.1.1 Attributs hérités

Cette classe ne possède aucun attribut hérité.

49.1.2 Attributs propres échangés

Cette classe définit l'attribut échangé suivant:

ElementaryAction Liste d'actions élémentaires incluses

(Action élémentaire)

49.2 Attributs internes propres

Cette classe ne définit aucun attribut interne propre.

49.3 Description formelle

```
Action Class      --> ElementaryAction+
ElementaryAction  --> Activate | Add | AddItem | Append | BringToFront |
                      Call | CallActionSlot | Clear | Clone |
                      CloseConnection | Deactivate | DelItem | Deselect |
                      DeselectItem | Divide | DrawArc | DrawLine | DrawOval
                      | DrawPolygon | DrawPolyline | DrawRectangle
                      |DrawSector | Fork | GetAvailabilityStatus |
                      GetBoxSize | GetCellItem | GetCursorPosition |
                      GetEngineSupport | GetEntryPoint | GetFillColour |
                      GetFirstItem | GetHighlightStatus |
                      GetInteractionStatus | GetItemStatus | GetLabel |
                      GetLastAnchorFired | GetLineColour | GetLineStyle |
                      GetLineWidth | GetListItem | GetListSize |
                      GetOverwriteMode | GetPortion | GetPosition |
                      GetRunningStatus | GetSelectionStatus | GetSliderValue
                      | GetTextContent | GetTextData | GetTokenPosition |
                      GetVolume | Launch | LockScreen | Modulo | Move |
                      MoveTo | Multiply | OpenConnection | Preload |
                      PutBefore | PutBehind | Quit | ReadPersistent | Run |
                      ScaleBitmap | ScaleVideo | ScrollItems | Select |
                      SelectItem | SendEvent | SendToBack | SetBoxSize |
                      SetCachePriority | SetCounterEndPosition |
                      SetCounterPosition | SetCounterTrigger |
                      SetCursorPosition | SetCursorShape | SetData |
                      SetEntryPoint | SetFillColour | SetFirstItem |
                      SetFontRef | SetHighlightStatus | SetInteractionStatus
                      | SetLabel | SetLineColour | SetLineStyle |
                      SetLineWidth | SetOverwriteMode | SetPaletteRef |
                      SetPortion | SetPosition | SetSliderValue | SetSpeed |
                      SetTimer | SetTransparency | SetVariable | SetVolume |
                      Spawn | Step | Stop | StorePersistent | Subtract |
                      TestVariable | Toggle | ToggleItem | TransitionTo |
                      Unload | UnlockScreen
```

NOTE – La sémantique et la syntaxe des actions sont fournies dans les paragraphes précédents de la présente Recommandation.

50 Références à Objets, Contenus, Valeurs, Couleurs et Position en X et Y

50.1 Référence à objet

Description

Ce type de donnée est utilisé pour référencer des objets. Les objets référencés seront visibles de l'objet duquel provient la référence. Ceci signifie que l'objet sera soit un objet *Scene* ou un objet *Application*, ou sera partie de l'application ou de la scène active.

La référence comprend un identificateur *GroupIdentifier* optionnel et un numéro d'objet *ObjectNumber*. La valeur par défaut du *GroupIdentifier* est celle du *GroupIdentifier* de l'objet *Scene* ou *Application* à partir duquel la référence a été établie.

```
ObjectReference             --> GroupIdentifier?,
                                ObjectNumber
GroupIdentifier             --> OctetString□
ObjectNumber                --> INTEGER□
```

50.2 Référence à contenu

Description

Ce type de donnée est utilisé pour référencer des sources de données externes. La référence *ContentReference* est composée d'un *OctetString*.

```
ContentReference            --> OctetString
```

50.3 Référence à objet générique

Description

Type de donnée permettant d'établir une référence directe à un objet ou une référence indirecte via un objet *Variable*.

Dans le cas d'une référence directe, cette référence résout une référence à objet directement vers l'objet cible.

Dans le cas de référence indirecte, cette référence résout une référence à objet vers un objet *ObjetRefVariable*. Celui-ci contiendra soit une référence à objet à destination de l'objet cible ou la valeur NULL.

```
GenericObjectReference      --> DirectReference |
                                IndirectReference
DirectReference             --> ObjectReference
IndirectReference           --> ObjectReference
```

50.4 Référence à contenu générique

Description — Type de donnée permettant d'établir une référence directe à une source de donnée externe ou une référence indirecte via un objet *Variable*.

Dans le cas d'une référence directe, cette référence résout une référence à contenu directement vers l'objet cible.

Dans le cas de référence indirecte, cette référence résout une référence à objet vers un objet *ContentRefVariable*. Celui-ci contiendra soit une référence à contenu à destination de l'objet cible ou la valeur NULL.

```
GenericContentReference     --> DirectContentReference     |
                                IndirectReference
DirectContentReference      --> ContentReference
IndirectReference           --> ObjectReference
```

50.5 Entier générique

Description — Type de donnée permettant soit d'insérer directement d'un entier ou de référencer un objet *IntegerVariable*.

```
GenericInteger              --> Value | IndirectReference
Value                       --> INTEGER
IndirectReference           --> ObjectReference
```

50.6 Booléen générique

Description — Type de donnée permettant d'insérer directement d'un booléen ou de référencer un objet *BooleanVariable*.

```
GenericBoolean              --> Value | IndirectReference
Value                       --> BOOLEAN
IndirectReference           --> ObjectReference
```

50.7 *OctetString* générique

Description — Type de donnée permettant d'insérer directement d'un *OctetString* ou de référencer un objet *OctetStringVariable*.

```
GenericOctetString          --> Value | IndirectReference
Value                       --> OctetString
IndirectReference           --> ObjectReference
```

50.8 Couleur

Description — Type de donnée utilisé pour spécifier une couleur par un nom (un *OctetString*) ou un index (un Entier référençant un objet *Palette*).

```
Colour                  --> ColourIndex | AbsoluteColour
ColourIndex             --> INTEGER
AbsoluteColour          --> OctetString
```

50.9 Position (X,Y)

Description — Type de donnée utilisée pour spécifier une position (X,Y) dans le système de coordonnées d'une scène.

```
XYPosition              --> XPosition,
                            YPosition
XPosition               --> INTEGER
YPosition               --> INTEGER
```

50.10 Résolution des valeurs génériques

Les valeurs génériques (*GenericContentReference*, *GenericObjectReference*, *GenericInteger*, *GenericBoolean*, et *GenericOctetString*) sont utilisées seulement comme paramètres d'actions élémentaires. La résolution a lieu au moment de l'invocation de l'action . Par exemple, considérons une variable entière V initialement mise à 10. Si les actions suivantes sont invoquées:

1) mettre V à 15;
2) mettre V comme valeur d'un curseur;
3) mettre V à 20,

la valeur du curseur sera mise à 15. En d'autres termes, la valeur du curseur est mise par valeur et non par référence.

51 Références aux objets MHEG-5

Les références aux objets MHEG-5 sont représentées par des *ObjectReferences*. A tout moment une référence *ObjectReference* sera résolue en prenant en compte à la fois l'objet MHEG-5 du *Group* contenant cette référence (une *Scene* ou une *Application*) et le contenu des attributs *GroupIdentifier* et *ObjectNumber* de l'attribut *ObjectReference*.

Ce qui suit présente comment *ObjectReference* sera codé selon le groupe *Group* d'origine et la nature de l'objet MHEG-5 référencé:

i) à l'intérieur d'un objet *Scene*
   a) référence à un *Ingredient* de la *Scene* active:
      1) le *GroupIdentifier* peut être ou ne pas être codé;
      2) *ObjectNumber* contiendra le numéro d'objet de l'*Ingredient* à l'intérieur de *Scene*.

b) référence à un *Ingredient* partagé de l'*Application* active:

1) *GroupIdentifier* contiendra l'identificateur de groupe de l'objet *Application* actif;

2) *ObjectNumber* contiendra le numéro d'objet de l'*Ingredient* à l'intérieur de l'objet *Application*.

c) référence à la *Scene* active elle-même:

1) le *GroupIdentifier* n'a pas besoin d'être codé;

2) *ObjectNumber* sera mis à 0.

d) référence à une autre *Scene*:

1) *GroupIdentifier* contiendra l'identificateur de groupe de l'objet *Scene* référencé;

2) *ObjectNumber* sera mis à 0.

e) référence à l'*Application* active:

1) *GroupIdentifier* contiendra l'identificateur de groupe de l'objet *Application* actif;

2) *ObjectNumber* sera mis à 0.

f) référence à un autre objet *Application*:

1) *GroupIdentifier* contiendra l'identificateur de groupe de l'objet *Application* référencé;

2) *ObjectNumber* sera mis à 0.

ii) à l'intérieur d'un objet *Application*

a) référence à un *Ingredient* du groupe *Application*:

1) le *GroupIdentifier* peut être ou ne pas être codé;

2) *ObjectNumber* contiendra le numéro d'objet unique de l'objet *Ingredient* à l'intérieur de l'objet *Application*.

b) référence à une *Scene*:

1) *GroupIdentifier* contiendra l'identificateur de groupe de l'objet *Scene* référencé;

2) *ObjectNumber* sera mis à 0.

c) référence à *l'Application* elle-même:

1) le *GroupIdentifier* n'a pas besoin d'être codé;

2) *ObjectNumber* sera mis à 0.

d) référence à un autre objet *Application*:

1) *GroupIdentifier* contiendra l'identificateur de groupe de l'objet *Application* référencé;

2) *ObjectNumber* sera mis à 0.

La présente Recommandation ne définit pas le codage réel de l'*OctetString* du *GroupIdentifier*. Chaque domaine applicatif MHEG-5 en définira des formes spécifiques.

52 Espaces de noms, Appels de Programmes distants et Connexions

Le moteur MHEG-5 possède un espace de nom par défaut qui est l'espace de nom de l'objet *Application*. Tous les autres objets *Application* pouvant être atteints à partir de cet espace (par *Launch* et *Spawn*) seront aussi dans cet espace. Cet espace de nom, au sens large, devrait aussi inclure assez d'information pour traiter les appels distants en direction de méthodes nommées à travers les objets *Program* MHEG-5.

NOTE – Dans l'Appendice I, on pourra trouver une explication sur la manière dont le moteur MHEG-5 se lie à cet espace de nom.

Cependant, il est aussi possible à une application MHEG-5 de se lier temporairement à un autre espace de nom, en utilisant l'action *OpenConnection*. Ce nouvel espace de nom peut être utilisé pour des appels de méthodes nommées (à travers les objets *Program* MHEG-5) dans le but d'accéder à un objet *Scene* MHEG-5 situé dans un espace de nom autre que celui de son objet *Application*. Les règles suivantes s'appliquent aux références faites ou en provenance d'un tel objet *Scene*:

1) la référence à un objet *Scene* dans un autre espace de nom que celui de l'objet *Application* sera codée à l'intérieur d'une action *TransitionTo* dont le paramètre *ConnectionTag* indique la connexion avec l'entité administrant cet autre espace de nom;

2) tous les *ContentReferences* à partir de cet objet *Scene* seront interprétés dans l'autre espace de nom;

3) toutes les références à *GroupIdentifier* à partir de cet objet *Scene* seront interprétées dans l'espace de nom par défaut, sauf comme indiqué dans la règle 1) ci-dessus.

Pour mémoire: l'implémentation dans le moteur MHEG-5 des actions *OpenConnection* et *CloseConnection* est une décision du domaine applicatif.

53 Gestion d'événement

53.1 Types d'événements

La présente Recommandation définit les événements suivants: *IsAvailable, ContentAvailable, IsDeleted, IsRunning, IsStopped, TestEvent, UserInput, TimerFired, AsynchStopped, InteractionCompleted, TokenMovedFrom, TokenMovedTo, FirstItemPresented, LastItemPresented, HeadItems, TailItems, ItemSelected, ItemDeselected, StreamEvent, StreamPlaying, StreamStopped, CounterTrigger, HighlightOn, HighlightOff, CursorEnter, CursorLeave, AnchorFired, IsSelected, IsDeselected, EntryFieldFull, EngineEvent.*

Un événement émane toujours d'un objet spécifique appelé la source de l'événement. La sémantique de chaque classe MHEG-5 définit les circonstances dans lesquelles un objet de cette classe génère un événement spécifique.

Certains des types d'événements ci-dessus ont une valeur de donnée associée. Cette valeur est utilisée pour déterminer si le l'objet *Link* associé devrait s'amorcer ou non, comme décrit ci-dessous. Le tableau suivant dresse une liste des valeurs de donnée associée à chaque type d'événement:

Type d'événement	Donnée associée	Type de valeur associée
AnchorFired	*AnchorTag*	*OctetString*
CounterTrigger	*Identifier*	*INTEGER*
EngineEvent	*EventTag*	*INTEGER*
FirstItemPresented	*Index*	*BOOLEAN*
HeadItems	Nombre d'éléments	*INTEGER*
ItemDeselected	*Index*	*INTEGER*
ItemSelected	*Index*	*INTEGER*
LastItemPresented	*Index*	*BOOLEAN*
StreamEvent	*StreamEventTag*	*OctetString*

Type d'événement	Donnée associée	Type de valeur associée
TailItems	*Number of items*	*INTEGER*
TestEvent	*TestResult*	*BOOLEAN*
TimerFired	*TimerIdentifier*	*INTEGER*
TokenMovedFrom	*Index*	*INTEGER*
TokenMovedTo	*Index*	*INTEGER*
UserInput	*UserInputEventTag*	*INTEGER*
Tous les autres	Aucune	Sans objet

Les événements dont les types sont associés à des données doivent toujours être générés avec cette donnée, par exemple lorsqu'un événement *TimerFired* est généré, il doit être accompagné par l'identificateur *TimerIdentifier*.

53.2 Evénements synchrones et asynchrones

Les événements peuvent apparaître pour deux raisons:

1) un processus externe au moteur MHEG-5 produit un événement.

   L'événement résultant est alors appelé asynchrone. Les événements asynchrones sont des événements du type *AnchorFired, AsynchStopped, ContentAvailable, CounterTrigger, CursorEnter, CursorLeave, EngineEvent, EntryFieldFull, InteractionCompleted, StreamEvent, StreamPlaying, StreamStopped, TimerFired and UserInput*;

2) l'événement est le résultat direct d'une action élémentaire.

   L'événement résultant est alors appelé synchrone. Les événements synchrones sont des événements du type *FirstItemPresented, HeadItems, HighlightOff, HighlightOn, IsAvailable, IsDeleted, IsDeselected, IsRunning, IsSelected, IsStopped, ItemDeselected, ItemSelected, LastItemPresented, TailItems, TestEvent, TokenMovedFrom and TokenMovedTo.* La sémantique des classes de la présente Recommandation fait état de manière explicite du type d'événement produit.

53.3 Gestion d'événements et liens

Chaque lien MHEG-5 possède une *LinkCondition* et un *LinkEffect*. Lorsque le moteur MHEG-5 examine un événement , il vérifiera tous les liens actifs (des objets *Scene* et *Application* actifs) pour voir si leurs attributs *EventType* et *EventSource* correspondent aux type et source de l'événement en question. Pour chacun des liens remplissant cette condition, la donnée associée à l'événement est vérifiée par rapport à l'attribut *EventData* optionnel du lien. Les liens remplissant cette condition aussi (ou la remplissant par défaut s'ils n'ont pas d'attribut *EventData*) sont dits *à amorcer.*

Le moteur MHEG-5 est piloté par l'apparition d'événements asynchrones. Sur occurrence d'un événement asynchrone, le moteur MHEG-5 examinera tous les liens actifs des objets *Scenes* et *Applications* actifs pour déterminer s'ils ont été amorcés. Pour chaque lien amorcé, les actions élémentaires de son *LinkEffect* seront stockées dans une file d'attente en vue d'une exécution séquentielle. La présente Recommandation *ne* spécifie *pas* l'ordre dans lequel deux liens s'amorçant sur un même événement seront traités.

Le traitement d'erreur par défaut est le suivant : si l'une des actions précédemment mentionnée produit une erreur, cette action élémentaire sera ignorée. Il est cependant autorisé pour un domaine applicatif d'utiliser des événements *EngineEvent* pour indiquer la situation d'erreur à l'application MHEG-5 (à savoir envoyer un message d'erreur).

En tant que résultat direct de l'exécution d'un *LinkEffect*, des événements synchrones peuvent apparaître. Ces événements seront traités directement par le moteur MHEG-5. En d'autres mots, à la suite de l'exécution de chaque action élémentaire, le moteur MHEG-5 vérifiera si d'éventuels liens supplémentaires ont été amorcés en tant que résultat de l'apparition d'événements synchrones. Si tel est le cas, ce lien et ses effets seront complètement traités avant que le moteur MHEG-5 ne traite l'action élémentaire suivante du lien d'origine.

Tout événement asynchrone (tel que l'événement *UserInput*) surgissant pendant que les événements asynchrones d'origine sont traités ne sont pas pris en compte tant que le processus ci-dessus n'est pas complètement terminé. Les événements asynchrones qui ne sont pas traités sont mis dans une file d'attente.

Les actions qui changent le contexte de traitement de l'action courante influenceront à la fois la file d'attente des événements asynchrones et celle des actions en cours d'attente de traitement. Le contexte est changé par les actions "*TransitionTo*", "*Launch*", "*Spawn*" et "*Quit*". Si une telle action apparaît, les événements asynchrones en attente dont l'origine est postérieure à la commutation de contexte seront enlevés de la file d'attente des événements asynchrones . Les actions élémentaires en attente d'exécution seront enlevées elles aussi de la file d'attente des actions.

Dans ce contexte, il faut noter que des actions MHEG-5, comme *Run*, ont un effet qui se prolonge au-delà de la fin de l'action elle-même. Par exemple, lorsqu'un objet *Bitmap* est exécuté par l'action *Run*, il rend la main aussitôt l'affichage de la *Bitmap* terminé, autorisant ainsi le processeur de lien à continuer son travail. L'effet de l'action (à savoir que la phototrame est sur l'écran) continue encore après la fin de l'action.

Un autre aspect important est qu'il est possible pour un lien de se désactiver lui-même par son *LinkEffect*. Une telle action sera différée jusqu'à ce que l'exécution du *LinkEffect* ait pris fin.

53.4 Saisie utilisateur

La présente Recommandation ne spécifie ni les dispositifs de saisie utilisateur, ni les types d'événements générés. La saisie " brute " de l'utilisateur reçue par le moteur MHEG -5 (par exemple sous la forme de commande de contrôle distante) sera traduite par le moteur, si approprié, en occurrences d'événements *UserInput*. Ces événements ont tous une balise spécifiant l'apparition de l'événement *UserInput*. La balise est un Entier, la sémantique de la balise est définie par un registre *InputEventRegister* attaché à l'objet *Scene*.

Une fois que le moteur MHEG-5 a traduit la saisie " brute " de l'utilisateur en un ou plusieurs événements *UserInput*, cet ou ces événements sont traités par le moteur comme décrit ci-dessus.

53.5 Interaction utilisateur

Les objets MHEG-5 appartenant à la classe *Interactible* peuvent être dans un certain état, appelé "*interacting*" qui est signalé par l'attribut *InteractionStatus* de l'objet mis à *True*. Lorsqu'un objet est dans cet état, aucun événement *UserInput* ne sera généré par le moteur MHEG-5. La raison à cette règle est que le moteur MHEG-5 pourrait avoir besoin d'utiliser la saisie "brute" de l'utilisateur afin d'interagir avec l'objet *Interactible*, et ensuite pourrait ne pas être capable de générer les événements *UserInput* normaux.

A tout moment, l'interaction portera au maximum sur un objet.

Cependant, tous les autres événements sont encore générés. Ceci rend possible, par exemple, l'implémentation de compteurs de dépassement de temps.

53.6 Evénements curseur

Les événements *CursorLeave* et *CursorEnter* définis précédemment nécessitent les précisions suivantes. Il peut s'avérer qu'un curseur se déplace d'une zone définie par un objet *Interactible* actif immédiatement dans une autre zone définie par un autre *Interactible*. Ceci peut être le cas, par exemple, lorsqu'une zone est sur l'autre dans la pile *DisplayStack* et ainsi la recouvre. Dans un tel cas, un événement *CursorLeave* sera généré par l'objet *Interactible* précédent (le plus haut des deux dans la pile *DisplayStack*), suivi par un événement *CursorEnter* généré par le dernier *Interactible* (le plus bas dans la pile *DisplayStack*).

Les événements curseur ont aussi à être générés si un objet *Interactible* est déplacé dessous la position du curseur, ou si un *Interactible* situé dessous la position du curseur devient actif, ou si l'ordre dans la pile des *Interactibles* situés dessous le curseur change d'une manière telle que *l'Interactible* le plus en haut situé sous le curseur vient à changer.

En général seul un objet *Interactible* actif à la fois peut avoir le curseur dans sa zone de délimitation. Ceci doit être reflété par des séquences adéquates d'événements *CursorLeave* et *CursorEnter*.

Finalement, il faut aussi rappeler ici qu'un objet *Interactible* inactif ne générera pas d'événement curseur.

53.7 Gestion d'erreur

D'une manière générale, le mécanisme d'erreur par défaut du moteur MHEG-5 consiste à ignorer la cause de l'erreur et à passer à l'étape suivante. Par exemple, si un objet échangé est une instance de classe non reconnue par le moteur MHEG-5, l'objet sera ignoré. Toute action ultérieure pouvant se produire en relation avec cet objet et produisant une erreur sera ignorée. Plus généralement, toute cause d'erreur (par exemple dans une action élémentaire) aura pour résultat d'ignorer cette cause (par exemple, l'action élémentaire).

Cependant, comme mentionné plus tôt, un domaine applicatif est autorisé à utilisé les événements *EngineEvent* pour indiquer la situation d'erreur à l'application MHEG-5 (par exemple, envoyer un message d'erreur).

54 Restitution des objets Visibles

54.1 Système de coordonnées

Le présent sous-paragraphe décrit la sémantique associée à la restitution d'objets de classes héritant de la classe *Visible*. Le moteur MHEG-5 aura accès à une seule zone d'affichage exactement, qui est appelée "l'écran" dans le reste du sous-paragraphe. L'écran est un système de coordonnées orthogonales avec une abscisse de la gauche vers la droite et une ordonnée du haut vers le bas. Son origine est le coin haut gauche. Selon les attributs *SceneCoordinateSystem* et *AspectRatio* de l'objet *Scene* actif en cours, le système de coordonnées peut avoir des tailles et des rapports d'aspect différents.

54.2 Zone de délimitation

Chaque objet *Visible* possède une zone de délimitation, spécifiée par les attributs internes *Position* et *BoxSize*. Ils sont tous les deux exprimés dans l'espace de coordonnée de la *Scene*. Les contenus de l'objet *Visible* seront ancrés au coin haut gauche de la zone de délimitation. De plus, les contenus de l'objet *Visible* seront tronqués de sorte que les parties tombant en dehors de la zone de délimitation ne soient pas affichées. La troncature sera effectuée de la manière suivante:

a) pour le texte et ses sous-classes, la troncature est effectuée de manière à respecter la position du texte à l'intérieur de la zone. Ici, seul le cas où *LineOrientation* vaut *horizontal* est expliqué ; le cas *vertical* est obtenue de manière analogue.

- Justification horizontale:

  si l'objet *Text* a l'attribut *TextWrapping* mis à *True*, son contenu est découpé en autant de lignes que nécessaire pour s'ajuster dans la zone de délimitation. De plus, si l'attribut *HorizontalJustification* de l'objet *Text* vaut *justified*, les lignes sont affichées de sorte que le début soit aligné sur la bordure gauche de la zone et la fin sur la bordure droite.

  Si l'attribut *TextWrapping* vaut *False*, les règles suivantes s'appliquent à la justification horizontale:

  – si *HorizontalJustification* vaut *start*, les lignes sont tronquées de sorte que chaque ligne soit alignée avec la bordure gauche de la zone de délimitation;

  – si *HorizontalJustification* vaut *end*, les lignes sont tronquées de sorte que chaque ligne soit alignée avec la bordure droite de la zone de délimitation;

  – si *HorizontalJustification* vaut *centre*, les lignes sont tronquées des deux côtés de sorte que chaque centre de ligne soit aligné sur une ligne verticale virtuelle passant au centre de la zone de délimitation.

- Justification verticale:

  pour un texte ayant l'attribut *VerticalJustification* mis à *end*, son contenu est tronqué de sorte que la fin du texte soit alignée avec le bas de la zone de délimitation ; ainsi, le début du texte est masqué. A l'opposé, si *VerticalJustification* vaut *start*, le texte est tronqué de sorte que sa fin soit masquée. Finalement si *VerticalJustification* vaut *centre*, le centre du texte est aligné avec le centre de la zone de délimitation, signifiant que le début et la fin sont tous les deux masqués.

- Couleur de fond:

  si le texte a une couleur de fond autre que transparent, sa zone de délimitation toute entière sera remplie avec cette couleur, que le texte remplisse ou non la zone de délimitation.

b) pour *Slider*, *Rectangle*, et les sous-classes de *Button*, l'objet n'est pas tronqué mais plutôt mis à l'échelle pour s'ajuster exactement à l'intérieur de la zone de délimitation;

c) les autres objets *Visibles* sont tronqués de sorte que leur coin haut gauche soit positionné au coin haut gauche de la zone de délimitation. Pour ces objets *Visibles*, une partie de la zone de délimitation peut rester vide, cette partie est alors transparente.

54.3 Pile de visualisation

Le moteur MHEG-5 implémentera une pile de visualisation. Chaque objet *Visible* est initialement restitué selon sa position dans l'attribut *DisplayStack* de l'objet *Application* en cours. Les objets *Visibles* peuvent être déplacés vers le haut et vers le bas dans la pile de visualisation au moyen des actions *BringToFront*, *SendToBack*, *PutBefore* et *PutBehind*. Lorsqu'un objet Visible est déplacé de cette manière, les (parties des) autres objets *Visibles* peuvent être masqués ou non. L'écran sera alors mis à jour conformément à ces déplacements.

54.4 Objets transparents

La transparence des objets est spécifiée au moyen d'un entier compris dans l'intervalle [0, 100] (pourcentage). L'interprétation de l'entier pour des valeurs allant de 1 à 99 dépend de l'implémentation, la présente Recommandation n'en donnant aucune définition. La présente Recommandation ne spécifie ni la précision réelle de la restitution et de l'affichage, ni les algorithmes utilisés pour effectuer ce travail. Cependant les valeurs de transparence allant de 0% à 100% sont spécifiées, elles sont définies comme suit.

Les objets de transparence 0% seront restitués comme des objets non transparents, à savoir les couleurs à l'origine seront restituées sans aucune modification. Les objets de transparence 100% seront restitués comme des objets transparents, à savoir les couleurs d'origine ne seront pas restituées et la couleur de fond sera perceptible.

i) pour chaque position dans l'espace de coordonnées de la *Scene*, au plus les *N* objets dont les zones de délimitation contiennent cette position sont considérés. Ces objets sont ordonnés dans la pile de visualisation. Appelons ces objets O_1, O_2, [...], O_N. où O_N représente l'objet le plus haut;

ii) le calcul de la valeur finale de pixel est faite à partir du bas de la pile. Cette valeur de pixel après la prise en compte de *j* objets est appelée V_j. La valeur finale de pixel est alors appelée V_N;

iii) le calcul de la valeur de pixel commence à partir d'un fond complètement noir. Donc, V_0 = <black>;

iv) pour *j* allant de 1 à *N*, les opérations suivantes sont effectuées:

a) si la valeur de pixel de O_j à considérer est appelée P_j et est *p*% translucide:

$$V_j = V_{j-1}\frac{P}{100} + P_j\left(1 - \frac{P}{100}\right).$$

Cette formule peut être appliquée à tous les composants de couleur d'un pixel si l'espace de couleur n'est pas basé sur des signaux de différence de couleur et de luminance (RGB dans un tel cas). Cependant, cette formule peut être appliquée à la luminance d'un pixel seulement si l'espace de couleur est basé sur des signaux de différence de couleur et de luminance tels que YUV. De plus les valeurs dans cette formule sont toutes supposées être exprimées en un seul même espace de couleurs linéaire. Si un espace de couleur non linéaire est utilisé, cette formule peut être prise en compte pour une restitution précise;

b) tout spécialement, si le pixel est complètement transparent (invisible):

$$V_j = V_{j-1}.$$

c) tout spécialement, si le pixel est complètement opaque:

$$V_j = P_j.$$

Ce modèle n'a pas pour but de contraindre un quelconque procédé de restitution et/ou d'affichage spécifique d'une implémentation de moteur MHEG-5 spécifique. Son seul objectif est de définir de manière non ambiguë ce que sont les sémantiques liées aux combinaisons de différents objets graphiques transparents. La présente Recommandation ne définit ni la précision réelle de la restitution et de l'affichage, ni les algorithmes utilisés pour effectuer ce travail.

54.5 Rapport d'aspect de pixel

Généralement , le rapport d'aspect de pixel d'une *Scene* et son contenu sont supposés être les mêmes, un pixel du contenu de la *Scene* étant alors mappé exactement sur un pixel de la *Scene*, aucune mise à l'échelle n'étant nécessaire. Si cependant les deux rapports d'aspect sont différents, le moteur MHEG-5 peut mettre à l'échelle le contenu, en prenant en compte les deux rapports d'aspect, afin de compenser la présentation du contenu. Il faut noter que cette mise à l'échelle est laissée à l'initiative de l'implémentation et est donc ainsi un attribut optionnel. La présente Recommandation ne définit aucun moyen pour le faire.

ANNEXE A

Notation ASN.1

La présente annexe décrit la notation ASN.1 pour la syntaxe des objets MHEG-5 conformes à l'ISO/CEI 13522-5.

Le codage des objets MHEG-5 à partir de cette syntaxe ASN.1 utilisera les *Règles de Codage Distinctives* définies dans la Rec. UIT-T X.690 | ISO/CEI 8825-1. Le codage alternatif est la notation textuelle définie dans l'Annexe B.

La syntaxe qui sera utilisée est détaillée ci-après:

```
-- $PREFIX=ISOMHEG-mheg-5:mheg-5
-- Module: mheg-5

ISO13522-MHEG-5 {joint-iso-itu-t(2) mheg(19) version(1) mheg-5(17)}

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- This module defines the MHEG-5 abstract syntax which consists of data values of type:
-- ISO13522-MHEG-5.InterchangedObject.
-- This abstract syntax is identified by the name: {joint-iso-itu-t(2) mheg(19) version(1) mheg-5(17)}.

InterchangedObject ::= CHOICE
{
  application [0] ApplicationClass,
  scene [1] SceneClass
}

-- A.1 Root Class ____________________________________________________

RootClass ::= ObjectReference

-- A.2 Group Class ___________________________________________________
```

```
GroupClass ::= SET
{
  RootClass (WITH COMPONENTS
    {external-reference (WITH COMPONENTS {..., object-number (0)}) PRESENT,
     internal-reference ABSENT}),
  standard-identifier [2] StandardIdentifier OPTIONAL,
  standard-version [3] INTEGER (1) OPTIONAL,
  object-information [4] OCTET STRING OPTIONAL,
  on-start-up [5] ActionClass OPTIONAL,
  on-close-down [6] ActionClass OPTIONAL,
  original-group-cache-priority [7] INTEGER (0..255) DEFAULT 127,
  items [8] SEQUENCE SIZE (1..MAX) OF GroupItem OPTIONAL
}

StandardIdentifier ::= SEQUENCE
{
  joint-iso-itu-t INTEGER (2),
  mheg INTEGER (19)
}

GroupItem ::= CHOICE
{
  resident-program [9] ResidentProgramClass,
  remote-program [10] RemoteProgramClass,
  interchanged-program [11] InterchangedProgramClass,
  palette [12] PaletteClass,
  font [13] FontClass,
  cursor-shape [14] CursorShapeClass,
  boolean-variable [15] BooleanVariableClass,
  integer-variable [16] IntegerVariableClass,
  octet-string-variable [17] OctetStringVariableClass,
  object-ref-variable [18] ObjectRefVariableClass,
  content-ref-variable [19] ContentRefVariableClass,
  link [20] LinkClass,
  stream [21] StreamClass,
  bitmap [22] BitmapClass,
  line-art [23] LineArtClass,
  dynamic-line-art [24] DynamicLineArtClass,
  rectangle [25] RectangleClass,
  hotspot [26] HotspotClass,
  switch-button [27] SwitchButtonClass,
  push-button [28] PushButtonClass,
  text [29] TextClass,
  entry-field [30] EntryFieldClass,
  hyper-text [31] HyperTextClass,
  slider [32] SliderClass,
  token-group [33] TokenGroupClass,
  list-group [34] ListGroupClass
}

-- A.3 Application Class ____________________________________________

ApplicationClass ::= SET
{
  COMPONENTS OF GroupClass,
  on-spawn-close-down [35] ActionClass OPTIONAL,
  on-restart [36] ActionClass OPTIONAL,
  default-attributes [37] SEQUENCE SIZE (1..MAX) OF DefaultAttribute OPTIONAL
}
```

```
DefaultAttribute ::= CHOICE
{
  character-set [38] INTEGER,
  background-colour [39] Colour,
  text-content-hook [40] INTEGER,
  text-colour [41] Colour,
  font [42] FontBody,
  font-attributes [43] OCTET STRING,
  interchanged-program-content-hook [44] INTEGER,
  stream-content-hook [45] INTEGER,
  bitmap-content-hook [46] INTEGER,
  line-art-content-hook [47] INTEGER,
  button-ref-colour [48] Colour,
  highlight-ref-colour [49] Colour,
  slider-ref-colour [50] Colour
}

FontBody ::= CHOICE
{
  direct-font OCTET STRING,
  indirect-font ObjectReference
}

-- A.4 Scene Class _____________________________________________________

SceneClass ::= SET
{
  COMPONENTS OF GroupClass,
  input-event-register [51] INTEGER,
  scene-coordinate-system [52] SceneCoordinateSystem,
  aspect-ratio [53] AspectRatio DEFAULT {width 4, height 3},
  moving-cursor [54] BOOLEAN DEFAULT FALSE,
  next-scenes [55] SEQUENCE SIZE (1..MAX) OF NextScene OPTIONAL
}

SceneCoordinateSystem ::= SEQUENCE
{
  x-scene INTEGER,
  y-scene INTEGER
}

AspectRatio ::= SEQUENCE
{
  width INTEGER,
  height INTEGER
}

NextScene ::= SEQUENCE
{
  scene-ref OCTET STRING,
  scene-weight INTEGER (0..255)
}
-- A.5 Ingredient Class _____________________________________________

IngredientClass ::= SET
{
  RootClass (WITH COMPONENTS
    {..., external-reference (WITH COMPONENTS {..., object-number (1..MAX)})}),
  initially-active [56] BOOLEAN DEFAULT TRUE,
  content-hook [57] INTEGER OPTIONAL,
```

```
  original-content [58] ContentBody OPTIONAL,
  shared [59] BOOLEAN DEFAULT FALSE
}

ContentBody ::= CHOICE
{
  included-content OCTET STRING,
  referenced-content ReferencedContent
}

ReferencedContent ::= SEQUENCE
{
  content-reference ContentReference,
  content-size [60] INTEGER OPTIONAL,
  content-cache-priority [61] INTEGER (0..255) DEFAULT 127
}

-- A.6 Link Class ___________________________________________________

LinkClass ::= SET
{
  COMPONENTS OF IngredientClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  link-condition [62] LinkCondition,
  link-effect [63] ActionClass
}

LinkCondition ::= SEQUENCE
{
  event-source ObjectReference,
  event-type EventType,
  event-data EventData OPTIONAL
}

EventType ::= ENUMERATED
{
  is-available(1),
  content-available(2),
  is-deleted(3),
  is-running(4),
  is-stopped(5),
  user-input(6),
  anchor-fired(7),
  timer-fired(8),
  asynch-stopped(9),
  interaction-completed(10),
  token-moved-from(11),
  token-moved-to(12),
  stream-event(13),
  stream-playing(14),
  stream-stopped(15),
  counter-trigger(16),
  highlight-on(17),
  highlight-off(18),
  cursor-enter(19),
  cursor-leave(20),
  is-selected(21),
  is-deselected(22),
  test-event(23),
  first-item-presented(24),
  last-item-presented(25),
```

```
  head-items(26),
  tail-items(27),
  item-selected(28),
  item-deselected(29),
  entry-field-full(30),
  engine-event(31)
}

EventData ::= CHOICE
{
  octetstring OCTET STRING,
  boolean BOOLEAN,
  integer INTEGER
}

-- A.7 Program Class _____________________________________________________

ProgramClass ::= SET
{
  COMPONENTS OF IngredientClass
    (WITH COMPONENTS {..., initially-active (FALSE) PRESENT}),
  name [64] OCTET STRING,
  initially-available [65] BOOLEAN DEFAULT TRUE
}

-- A.8 Resident Program Class ____________________________________________

ResidentProgramClass ::= ProgramClass
  (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT})

-- A.9 Remote Program Class _____________________________________________

RemoteProgramClass ::= SET
{
  COMPONENTS OF ProgramClass
  (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  program-connection-tag [66] INTEGER OPTIONAL
}

-- A.10 Interchanged Program Class _____________________________________

InterchangedProgramClass ::= ProgramClass
  (WITH COMPONENTS {..., original-content PRESENT})

-- A.11 Palette Class ___________________________________________________

PaletteClass ::= IngredientClass
  (WITH COMPONENTS
    {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})

-- A.12 Font Class ______________________________________________________

FontClass ::= IngredientClass
  (WITH COMPONENTS
    {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})

-- A.13 Cursor Shape ____________________________________________________

CursorShapeClass ::= IngredientClass
  (WITH COMPONENTS
    {..., content-hook PRESENT, original-content PRESENT, initially-active (TRUE)})
```

```
-- A.14 Variable Class _____________________________________________

VariableClass ::= SET
{
  COMPONENTS OF IngredientClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT, initially-active (TRUE)}),
  original-value [67] OriginalValue
}

OriginalValue ::= CHOICE
{
  boolean BOOLEAN,
  integer INTEGER,
  octetstring OCTET STRING,
  object-reference [68] ObjectReference,
  content-reference [69] ContentReference
}

-- A.15 Boolean Variable Class _____________________________________

BooleanVariableClass ::= VariableClass
  (WITH COMPONENTS {..., original-value (WITH COMPONENTS {..., boolean PRESENT})})

-- A.16 Integer Variable Class _____________________________________

IntegerVariableClass ::= VariableClass
  (WITH COMPONENTS {..., original-value (WITH COMPONENTS {..., integer PRESENT})})

-- A.17 Octet String Variable Class ________________________________

OctetStringVariableClass ::= VariableClass
  (WITH COMPONENTS {..., original-value (WITH COMPONENTS {..., octetstring PRESENT})})

-- A.18 Object Reference Variable Class ____________________________

ObjectRefVariableClass ::= VariableClass
  (WITH COMPONENTS {..., original-value (WITH COMPONENTS {..., object-reference PRESENT})})

-- A.19 Content Reference Variable Class ___________________________

ContentRefVariableClass ::= VariableClass
  (WITH COMPONENTS {..., original-value (WITH COMPONENTS {..., content-reference PRESENT})})

-- A.20 Presentable Class __________________________________________

PresentableClass ::= IngredientClass

-- A.21 Token Manager Class ________________________________________

TokenManagerClass ::= SET
{
  movement-table [70] SEQUENCE SIZE (1..MAX) OF Movement OPTIONAL
}

Movement ::= SEQUENCE SIZE (1..MAX) OF INTEGER

-- A.22 Token Group Class __________________________________________

TokenGroupClass ::= SET
{
```

```
  COMPONENTS OF PresentableClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  COMPONENTS OF TokenManagerClass,
  token-group-items [71] SEQUENCE SIZE (1..MAX) OF TokenGroupItem,
  no-token-action-slots [72] SEQUENCE SIZE (1..MAX) OF ActionSlot OPTIONAL
}

TokenGroupItem ::= SEQUENCE
{
  a-visible ObjectReference,
  action-slots SEQUENCE SIZE (1..MAX) OF ActionSlot OPTIONAL
}

ActionSlot ::= CHOICE
{
  action-class ActionClass,
  null NULL
}

-- A.23 List Group Class ___________________________________________

ListGroupClass ::= SET
{
  COMPONENTS OF TokenGroupClass,
  positions [73] SEQUENCE SIZE (1..MAX) OF XYPosition,
  wrap-around [74] BOOLEAN DEFAULT FALSE,
  multiple-selection [75] BOOLEAN DEFAULT FALSE
}

-- A.24 Visible Class ______________________________________________

VisibleClass ::= SET
{
  COMPONENTS OF PresentableClass,
  original-box-size [76] OriginalBoxSize,
  original-position [77] XYPosition DEFAULT {x-position 0, y-position 0},
  original-palette-ref [78] ObjectReference OPTIONAL
}

OriginalBoxSize ::= SEQUENCE
{
  x-length INTEGER (0..MAX),
  y-length INTEGER (0..MAX)
}

-- A.25 Bitmap Class _______________________________________________

BitmapClass ::= SET
{
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., original-content PRESENT}),
  tiling [79] BOOLEAN DEFAULT FALSE,
  original-transparency [80] INTEGER (0..100) DEFAULT 0
}

-- A.26 Line Art Class _____________________________________________

LineArtClass ::= SET
{
```

```
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., original-content PRESENT}),
  bordered-bounding-box [81] BOOLEAN DEFAULT TRUE,
  original-line-width [82] INTEGER DEFAULT 1,
  original-line-style [83] INTEGER {solid(1), dashed(2), dotted(3)} DEFAULT solid,
  original-ref-line-colour [84] Colour OPTIONAL,
  original-ref-fill-colour [85] Colour OPTIONAL
}

-- A.27 Rectangle Class ____________________________________________

RectangleClass ::= LineArtClass
  (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT, bordered-bounding-box ABSENT})

-- A.28 Dynamic Line Art Class _____________________________________

DynamicLineArtClass ::= LineArtClass
  (WITH COMPONENTS
    {..., content-hook ABSENT, original-content ABSENT})

-- A.29 Text Class _________________________________________________

TextClass ::= SET
{
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., original-content PRESENT}),
  original-font [86] FontBody OPTIONAL,
  font-attributes [43] OCTET STRING OPTIONAL,
  text-colour [41] Colour OPTIONAL,
  background-colour [39] Colour OPTIONAL,
  character-set [38] INTEGER OPTIONAL,
  horizontal-justification [87] Justification DEFAULT start,
  vertical-justification [88] Justification DEFAULT start,
  line-orientation [89] LineOrientation DEFAULT horizontal,
  start-corner [90] StartCorner DEFAULT upper-left,
  text-wrapping [91] BOOLEAN DEFAULT FALSE
}

Justification ::= ENUMERATED
{
  start(1),
  end(2),
  centre(3),
  justified(4)
}

LineOrientation ::= ENUMERATED {vertical(1), horizontal(2)}

StartCorner ::= ENUMERATED
{
  upper-left(1),
  upper-right(2),
  lower-left(3),
  lower-right(4)
}

-- A.30 Stream Class ____________________________________________

StreamClass ::= SET
{
```

```
  COMPONENTS OF PresentableClass
    (WITH COMPONENTS {..., original-content PRESENT}),
  multiplex [92] SEQUENCE SIZE (1..MAX) OF StreamComponent,
  storage [93] Storage DEFAULT stream,
  looping [94] INTEGER {infinity(0)} DEFAULT 1
}

StreamComponent ::= CHOICE
{
  audio [95] AudioClass,
  video [96] VideoClass,
  rtgraphics [97] RTGraphicsClass
}

Storage ::= ENUMERATED {memory(1), stream(2)}

-- A.31 Audio Class ____________________________________________________

AudioClass ::= SET
{
  COMPONENTS OF PresentableClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT, shared ABSENT}),
  component-tag [98] INTEGER,
  original-volume [99] INTEGER DEFAULT 0
}

-- A.32 Video Class ____________________________________________________

VideoClass ::= SET
{
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT, shared ABSENT, original-
palette-ref ABSENT}),
  component-tag [98] INTEGER,
  termination [100] Termination DEFAULT disappear
}

Termination ::= ENUMERATED {freeze(1), disappear(2)}

-- A.33 RTGraphics Class _____________________________________________

RTGraphicsClass ::= SET
{
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT, shared ABSENT}),
  component-tag [98] INTEGER,
  termination [100] Termination DEFAULT disappear
}

-- A.34 Interactible Class ____________________________________________

InteractibleClass ::= SET
{
  engine-resp [101] BOOLEAN DEFAULT TRUE,
  highlight-ref-colour [49] Colour OPTIONAL
}

-- A.35 Slider Class ____________________________________________________

SliderClass ::= SET
{
```

```
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  COMPONENTS OF InteractibleClass,
  orientation [102] Orientation,
  max-value [103] INTEGER,
  min-value [104] INTEGER DEFAULT 1,
  initial-value [105] INTEGER OPTIONAL,
  initial-portion [106] INTEGER OPTIONAL,
  step-size [107] INTEGER DEFAULT 1,
  slider-style [108] SliderStyle DEFAULT normal,
  slider-ref-colour [50] Colour OPTIONAL
}

Orientation ::= ENUMERATED {left(1), right(2), up(3), down(4)}

SliderStyle ::= ENUMERATED {normal(1), thermometer(2), proportional(3)}

-- A.36 Entry Field Class ____________________________________________

EntryFieldClass ::= SET
{
  COMPONENTS OF TextClass,
  COMPONENTS OF InteractibleClass,
  input-type [109] InputType DEFAULT any,
  char-list [110] OCTET STRING OPTIONAL,
  obscured-input [111] BOOLEAN DEFAULT FALSE,
  max-length [112] INTEGER DEFAULT 0
}

InputType ::= ENUMERATED {alpha(1), numeric(2), any(3), listed(4)}

-- A.37 Hyper Text Class _____________________________________________

HyperTextClass ::= SET
{
  COMPONENTS OF TextClass,
  COMPONENTS OF InteractibleClass
}

-- A.38 Button Class _________________________________________________

ButtonClass ::= SET
{
  COMPONENTS OF VisibleClass
    (WITH COMPONENTS {..., content-hook ABSENT, original-content ABSENT}),
  COMPONENTS OF InteractibleClass,
  button-ref-colour [48] Colour OPTIONAL
}

-- A.39 Hotspot Class ________________________________________________

HotspotClass ::= ButtonClass

-- A.40 Push Button Class ____________________________________________

PushButtonClass ::= SET
{
  COMPONENTS OF ButtonClass,
  original-label [113] OCTET STRING OPTIONAL,
  character-set [38] INTEGER OPTIONAL
}
```

-- A.41 Switch Button Class _____

```
SwitchButtonClass ::= SET
{
  COMPONENTS OF PushButtonClass,
  button-style [114] ButtonStyle
}

ButtonStyle ::= ENUMERATED
{
  pushbutton(1),
  radiobutton(2),
  checkbox(3)
}
```

-- A.42 Action Class _____

```
ActionClass ::= SEQUENCE SIZE (1..MAX) OF ElementaryAction

ElementaryAction ::= CHOICE
{
  activate [115] GenericObjectReference,
  add [116] Add,
  add-item [117] AddItem,
  append [118] Append,
  bring-to-front [119] GenericObjectReference,
  call [120] Call,
  call-action-slot [121] CallActionSlot,
  clear [122] GenericObjectReference,
  clone [123] Clone,
  close-connection [124] CloseConnection,
  deactivate [125] GenericObjectReference,
  del-item [126] DelItem,
  deselect [127] GenericObjectReference,
  deselect-item [128] DeselectItem,
  divide [129] Divide,
  draw-arc [130] DrawArc,
  draw-line [131] DrawLine,
  draw-oval [132] DrawOval,
  draw-polygon [133] DrawPolygon,
  draw-polyline [134] DrawPolyline,
  draw-rectangle [135] DrawRectangle,
  draw-sector [136] DrawSector,
  fork [137] Fork,
  get-availability-status [138] GetAvailabilityStatus,
  get-box-size [139] GetBoxSize,
  get-cell-item [140] GetCellItem,
  get-cursor-position [141] GetCursorPosition,
  get-engine-support [142] GetEngineSupport,
  get-entry-point [143] GetEntryPoint,
  get-fill-colour [144] GetFillColour,
  get-first-item [145] GetFirstItem,
  get-highlight-status [146] GetHighlightStatus,
  get-interaction-status [147] GetInteractionStatus,
  get-item-status [148] GetItemStatus,
  get-label [149] GetLabel,
  get-last-anchor-fired [150] GetLastAnchorFired,
  get-line-colour [151] GetLineColour,
  get-line-style [152] GetLineStyle,
  get-line-width [153] GetLineWidth,
```

get-list-item [154] GetListItem,
get-list-size [155] GetListSize,
get-overwrite-mode [156] GetOverwriteMode,
get-portion [157] GetPortion,
get-position [158] GetPosition,
get-running-status [159] GetRunningStatus,
get-selection-status [160] GetSelectionStatus,
get-slider-value [161] GetSliderValue,
get-text-content [162] GetTextContent,
get-text-data [163] GetTextData,
get-token-position [164] GetTokenPosition,
get-volume [165] GetVolume,
launch [166] GenericObjectReference,
lock-screen [167] GenericObjectReference,
modulo [168] Modulo,
move [169] Move,
move-to [170] MoveTo,
multiply [171] Multiply,
open-connection [172] OpenConnection,
preload [173] GenericObjectReference,
put-before [174] PutBefore,
put-behind [175] PutBehind,
quit [176] GenericObjectReference,
read-persistent [177] ReadPersistent,
run [178] GenericObjectReference,
scale-bitmap [179] ScaleBitmap,
scale-video [180] ScaleVideo,
scroll-items [181] ScrollItems,
select [182] GenericObjectReference,
select-item [183] SelectItem,
send-event [184] SendEvent,
send-to-back [185] GenericObjectReference,
set-box-size [186] SetBoxSize,
set-cache-priority [187] SetCachePriority,
set-counter-end-position [188] SetCounterEndPosition,
set-counter-position [189] SetCounterPosition,
set-counter-trigger [190] SetCounterTrigger,
set-cursor-position [191] SetCursorPosition,
set-cursor-shape [192] SetCursorShape,
set-data [193] SetData,
set-entry-point [194] SetEntryPoint,
set-fill-colour [195] SetFillColour,
set-first-item [196] SetFirstItem,
set-font-ref [197] SetFontRef,
set-highlight-status [198] SetHighlightStatus,
set-interaction-status [199] SetInteractionStatus,
set-label [200] SetLabel,
set-line-colour [201] SetLineColour,
set-line-style [202] SetLineStyle,
set-line-width [203] SetLineWidth,
set-overwrite-mode [204] SetOverwriteMode,
set-palette-ref [205] SetPaletteRef,
set-portion [206] SetPortion,
set-position [207] SetPosition,
set-slider-value [208] SetSliderValue,
set-speed [209] SetSpeed,
set-timer [210] SetTimer,
set-transparency [211] SetTransparency,
set-variable [212] SetVariable,
set-volume [213] SetVolume,
spawn [214] GenericObjectReference,

```
  step [215] Step,
  stop [216] GenericObjectReference,
  store-persistent [217] StorePersistent,
  subtract [218] Subtract,
  test-variable [219] TestVariable,
  toggle [220] GenericObjectReference,
  toggle-item [221] ToggleItem,
  transition-to [222] TransitionTo,
  unload [223] GenericObjectReference,
  unlock-screen [224] GenericObjectReference
}

Add ::= SEQUENCE
{
  target GenericObjectReference,
  value GenericInteger
}

AddItem ::= SEQUENCE
{
  target GenericObjectReference,
  item-index GenericInteger,
  visible-reference GenericObjectReference
}

Append ::= SEQUENCE
{
  target GenericObjectReference,
  append-value GenericOctetString
}

Call ::= SEQUENCE
{
  target GenericObjectReference,
  call-succeeded ObjectReference,
  parameters SEQUENCE SIZE (1..MAX) OF Parameter OPTIONAL
}

CallActionSlot ::= SEQUENCE
{
  target GenericObjectReference,
  index GenericInteger
}

Clone ::= SEQUENCE
{
  target GenericObjectReference,
  clone-ref-var ObjectReference
}

CloseConnection ::= SEQUENCE
{
  target GenericObjectReference,
  connection-tag GenericInteger
}

DelItem ::= SEQUENCE
{
  target GenericObjectReference,
  visible-reference GenericObjectReference
}
```

```
DeselectItem ::= SEQUENCE
{
  target GenericObjectReference,
  item-index GenericInteger
}

Divide ::= SEQUENCE
{
  target GenericObjectReference,
  value GenericInteger
}

DrawArc ::= SEQUENCE
{
  target GenericObjectReference,
  x GenericInteger,
  y GenericInteger,
  ellipse-width GenericInteger,
  ellipse-height GenericInteger,
  start-angle GenericInteger,
  arc-angle GenericInteger
}

DrawLine ::= SEQUENCE
{
  target GenericObjectReference,
  x1 GenericInteger,
  y1 GenericInteger,
  x2 GenericInteger,
  y2 GenericInteger
}

DrawOval ::= SEQUENCE
{
  target GenericObjectReference,
  x GenericInteger,
  y GenericInteger,
  ellipse-width GenericInteger,
  ellipse-height GenericInteger
}

DrawPolygon ::= SEQUENCE
{
  target GenericObjectReference,
  pointlist SEQUENCE SIZE (1..MAX) OF Point
}

DrawPolyline ::= SEQUENCE
{
  target GenericObjectReference,
  pointlist SEQUENCE SIZE (1..MAX) OF Point
}

DrawRectangle ::= SEQUENCE
{
  target GenericObjectReference,
  x1 GenericInteger,
  y1 GenericInteger,
```

```
  x2 GenericInteger,
  y2 GenericInteger
}

DrawSector ::= SEQUENCE
{
  target GenericObjectReference,
  x GenericInteger,
  y GenericInteger,
  ellipse-width GenericInteger,
  ellipse-height GenericInteger,
  start-angle GenericInteger,
  arc-angle GenericInteger
}

Fork ::= SEQUENCE
{
  target GenericObjectReference,
  fork-succeeded ObjectReference,
  parameters SEQUENCE SIZE (1..MAX) OF Parameter OPTIONAL
}

GetAvailabilityStatus ::= SEQUENCE
{
  target GenericObjectReference,
  availability-status-var ObjectReference
}

GetBoxSize ::= SEQUENCE
{
  target GenericObjectReference,
  x-box-size-var ObjectReference,
  y-box-size-var ObjectReference
}

GetCellItem ::= SEQUENCE
{
  target GenericObjectReference,
  cell-index GenericInteger,
  item-ref-var ObjectReference
}

GetCursorPosition ::= SEQUENCE
{
  target GenericObjectReference,
  x-out ObjectReference,
  y-out ObjectReference
}

GetEngineSupport ::= SEQUENCE
{
  target GenericObjectReference,
  feature GenericOctetString,
  answer ObjectReference
}

GetEntryPoint ::= SEQUENCE
{
  target GenericObjectReference,
  entry-point-var ObjectReference
}
```

```
GetFillColour ::= SEQUENCE
{
  target GenericObjectReference,
  fill-colour-var ObjectReference
}

GetFirstItem ::= SEQUENCE
{
  target GenericObjectReference,
  first-item-var  ObjectReference
}

GetHighlightStatus ::= SEQUENCE
{
  target GenericObjectReference,
  highlight-status-var ObjectReference
}

GetInteractionStatus ::= SEQUENCE
{
  target GenericObjectReference,
  interaction-status-var ObjectReference
}

GetItemStatus ::= SEQUENCE
{
  target GenericObjectReference,
  item-index GenericInteger,
  item-status-var ObjectReference
}

GetLabel ::= SEQUENCE
{
  target GenericObjectReference,
  label-var ObjectReference
}

GetLastAnchorFired ::= SEQUENCE
{
  target GenericObjectReference,
  last-anchor-fired-var ObjectReference
}

GetLineColour ::= SEQUENCE
{
  target GenericObjectReference,
  line-colour-var ObjectReference
}

GetLineStyle ::= SEQUENCE
{
  target GenericObjectReference,
  line-style-var ObjectReference
}

GetLineWidth ::= SEQUENCE
{
  target GenericObjectReference,
  line-width-var ObjectReference
}
```

```
GetListItem ::= SEQUENCE
{
  target GenericObjectReference,
  item-index GenericInteger,
  item-ref-var ObjectReference
}

GetListSize ::= SEQUENCE
{
  target GenericObjectReference,
  size-var ObjectReference
}

GetOverwriteMode ::= SEQUENCE
{
  target GenericObjectReference,
  overwrite-mode-var ObjectReference
}

GetPortion ::= SEQUENCE
{
  target GenericObjectReference,
  portion-var ObjectReference
}

GetPosition ::= SEQUENCE
{
  target GenericObjectReference,
  x-position-var ObjectReference,
  y-position-var ObjectReference
}

GetRunningStatus ::= SEQUENCE
{
  target GenericObjectReference,
  running-status-var ObjectReference
}

GetSelectionStatus ::= SEQUENCE
{
  target GenericObjectReference,
  selection-status-var ObjectReference
}

GetSliderValue ::= SEQUENCE
{
  target GenericObjectReference,
  slider-value-var ObjectReference
}

GetTextContent ::= SEQUENCE
{
  target GenericObjectReference,
  text-content-var ObjectReference
}
```

```
GetTextData ::= SEQUENCE
{
  target GenericObjectReference,
  text-data-var ObjectReference
}

GetTokenPosition ::= SEQUENCE
{
  target GenericObjectReference,
  token-position-var ObjectReference
}

GetVolume ::= SEQUENCE
{
  target GenericObjectReference,
  volume-var ObjectReference
}

Modulo ::= SEQUENCE
{
  target GenericObjectReference,
  value GenericInteger
}

Move ::= SEQUENCE
{
  target GenericObjectReference,
  movement-identifier GenericInteger
}

MoveTo ::= SEQUENCE
{
  target GenericObjectReference,
  index GenericInteger
}

Multiply ::= SEQUENCE
{
  target GenericObjectReference,
  value GenericInteger
}

OpenConnection ::= SEQUENCE
{
  target GenericObjectReference,
  open-succeeded ObjectReference,
  protocol GenericOctetString,
  address GenericOctetString,
  connection-tag GenericInteger
}

PutBefore ::= SEQUENCE
{
  target GenericObjectReference,
  reference-visible GenericObjectReference
}
```

```
PutBehind ::= SEQUENCE
{
  target GenericObjectReference,
  reference-visible GenericObjectReference
}

ReadPersistent ::= SEQUENCE
{
  target GenericObjectReference,
  read-succeeded ObjectReference,
  out-variables SEQUENCE SIZE (1..MAX) OF ObjectReference,
  in-file-name GenericOctetString
}

ScaleBitmap ::= SEQUENCE
{
  target GenericObjectReference,
  x-scale GenericInteger,
  y-scale GenericInteger
}

ScaleVideo ::= SEQUENCE
{
  target GenericObjectReference,
  x-scale GenericInteger,
  y-scale GenericInteger
}

ScrollItems ::= SEQUENCE
{
  target GenericObjectReference,
  items-to-scroll GenericInteger
}

SelectItem ::= SEQUENCE
{
  target GenericObjectReference,
  item-index GenericInteger
}

SendEvent ::= SEQUENCE
{
  target GenericObjectReference,
  emulated-event-source GenericObjectReference,
  emulated-event-type EventType,
  emulated-event-data EmulatedEventData OPTIONAL
}

SetBoxSize ::= SEQUENCE
{
  target GenericObjectReference,
  x-new-box-size GenericInteger,
  y-new-box-size GenericInteger
}

SetCachePriority ::= SEQUENCE
{
  target GenericObjectReference,
  new-cache-priority GenericInteger
}
```

```
SetCounterEndPosition ::= SEQUENCE
{
  target GenericObjectReference,
  new-counter-end-position GenericInteger
}

SetCounterPosition ::= SEQUENCE
{
  target GenericObjectReference,
  new-counter-position GenericInteger
}

SetCounterTrigger ::= SEQUENCE
{
  target GenericObjectReference,
  trigger-identifier GenericInteger,
  new-counter-value GenericInteger OPTIONAL
}

SetCursorPosition ::= SEQUENCE
{
  target GenericObjectReference,
  x-cursor GenericInteger,
  y-cursor GenericInteger
}

SetCursorShape ::= SEQUENCE
{
  target GenericObjectReference,
  new-cursor-shape GenericObjectReference OPTIONAL
}

SetData ::= SEQUENCE
{
  target GenericObjectReference,
  new-content NewContent
}

SetEntryPoint ::= SEQUENCE
{
  target GenericObjectReference,
  new-entry-point GenericInteger
}

SetFillColour ::= SEQUENCE
{
  target GenericObjectReference,
  new-fill-colour NewColour OPTIONAL
}

SetFirstItem ::= SEQUENCE
{
  target GenericObjectReference,
  new-first-item GenericInteger
}

SetFontRef ::= SEQUENCE
{
  target GenericObjectReference,
  new-font NewFont
}
```

```
SetHighlightStatus ::= SEQUENCE
{
  target GenericObjectReference,
  new-highlight-status GenericBoolean
}

SetInteractionStatus ::= SEQUENCE
{
  target GenericObjectReference,
  new-interaction-status GenericBoolean
}

SetLabel ::= SEQUENCE
{
  target GenericObjectReference,
  new-label GenericOctetString
}

SetLineColour ::= SEQUENCE
{
  target GenericObjectReference,
  new-line-colour NewColour
}

SetLineStyle ::= SEQUENCE
{
  target GenericObjectReference,
  new-line-style GenericInteger
}

SetLineWidth ::= SEQUENCE
{
  target GenericObjectReference,
  new-line-width GenericInteger
}

SetOverwriteMode ::= SEQUENCE
{
  target GenericObjectReference,
  new-overwrite-mode GenericBoolean
}

SetPaletteRef ::= SEQUENCE
{
  target GenericObjectReference,
  new-palette-ref GenericObjectReference
}

SetPortion ::= SEQUENCE
{
  target GenericObjectReference,
  new-portion GenericInteger
}

SetPosition ::= SEQUENCE
{
  target GenericObjectReference,
  new-x-position GenericInteger,
  new-y-position GenericInteger
}
```

```
SetSliderValue ::= SEQUENCE
{
  target GenericObjectReference,
  new-slider-value GenericInteger
}

SetSpeed ::= SEQUENCE
{
  target GenericObjectReference,
  new-speed Rational
}

SetTimer ::= SEQUENCE
{
  target GenericObjectReference,
  timer-id GenericInteger,
  new-timer NewTimer OPTIONAL
}

NewTimer ::= SEQUENCE
{
  timer-value GenericInteger,
  absolute-time GenericBoolean OPTIONAL
}

SetTransparency ::= SEQUENCE
{
  target GenericObjectReference,
  new-transparency GenericInteger
}

SetVariable ::= SEQUENCE
{
  target GenericObjectReference,
  new-variable-value NewVariableValue
}

SetVolume ::= SEQUENCE
{
  target GenericObjectReference,
  new-volume GenericInteger
}

Step ::= SEQUENCE
{
  target GenericObjectReference,
  nb-of-steps GenericInteger
}

StorePersistent ::= SEQUENCE
{
  target GenericObjectReference,
  store-succeeded ObjectReference,
  in-variables SEQUENCE SIZE (1..MAX) OF ObjectReference,
  out-file-name GenericOctetString
}
```

```
Subtract ::= SEQUENCE
{
  target GenericObjectReference,
  value GenericInteger
}

TestVariable ::= SEQUENCE
{
  target GenericObjectReference,
  operator GenericInteger,
  comparison-value ComparisonValue
}

ToggleItem ::= SEQUENCE
{
  target GenericObjectReference,
  item-index GenericInteger
}

TransitionTo ::= SEQUENCE
{
  target GenericObjectReference,
  connection-tag-or-null ConnectionTagOrNull,
  transition-effect GenericInteger OPTIONAL
}

ConnectionTagOrNull ::= CHOICE
{
  connection-tag GenericInteger,
  null NULL
}

ComparisonValue ::= CHOICE
{
  new-generic-boolean [225] GenericBoolean,
  new-generic-integer [226] GenericInteger,
  new-generic-octetstring [227] GenericOctetString,
  new-generic-object-reference [228] GenericObjectReference,
  new-generic-content-reference [229] GenericContentReference
}

EmulatedEventData ::= CHOICE
{
  new-generic-boolean [225] GenericBoolean,
  new-generic-integer [226] GenericInteger,
  new-generic-octet-string [227] GenericOctetString
}

NewColour ::= CHOICE
{
  new-colour-index [230] GenericInteger,
  new-absolute-colour [231] GenericOctetString
}

NewContent ::= CHOICE
{
  new-included-content GenericOctetString,
  new-referenced-content NewReferencedContent
}
```

```
NewFont ::= CHOICE
{
  new-font-name [232] GenericOctetString,
  new-font-reference [233] GenericObjectReference
}

NewReferencedContent ::= SEQUENCE
{
  generic-content-reference GenericContentReference,
  new-content-size [234] NewContentSize,
  new-content-cache-priority [235] GenericInteger OPTIONAL
}

NewContentSize ::= CHOICE
{
  content-size GenericInteger,
  null NULL
}

NewVariableValue ::= CHOICE
{
  new-generic-integer [226] GenericInteger,
  new-generic-boolean [225] GenericBoolean,
  new-generic-octet-string [227] GenericOctetString,
  new-generic-object-reference [228] GenericObjectReference,
  new-generic-content-reference [229] GenericContentReference
}

Parameter ::= CHOICE
{
  new-generic-boolean [225] GenericBoolean,
  new-generic-integer [226] GenericInteger,
  new-generic-octetstring [227] GenericOctetString,
  new-generic-object-reference [228] GenericObjectReference,
  new-generic-content-reference [229] GenericContentReference
}

Point ::= SEQUENCE
{
  x GenericInteger,
  y GenericInteger
}

Rational ::= SEQUENCE
{
  numerator GenericInteger,
  denominator GenericInteger OPTIONAL
}

-- A.43 Referencing Objects, Contents, Values, Colour and Position ____

ObjectReference ::= CHOICE
{
  external-reference ExternalReference,
  internal-reference INTEGER (1..MAX)
}
```

```
ExternalReference ::= SEQUENCE
{
  group-identifier OCTET STRING,
  object-number INTEGER (0..MAX)
}

IndirectReference ::= [236] ObjectReference

ContentReference ::= OCTET STRING

GenericObjectReference ::= CHOICE
{
  direct-reference ObjectReference,
  indirect-reference IndirectReference
}

GenericContentReference ::= CHOICE
{
  content-reference [69] ContentReference,
  indirect-reference IndirectReference
}

GenericInteger ::= CHOICE
{
  integer INTEGER,
  indirect-reference IndirectReference
}

GenericBoolean ::= CHOICE
{
  boolean BOOLEAN,
  indirect-reference IndirectReference
}

GenericOctetString ::= CHOICE
{
  octetstring OCTET STRING,
  indirect-reference IndirectReference
}

Colour ::= CHOICE
{
  colour-index INTEGER,
  absolute-colour OCTET STRING
}

XYPosition ::= SEQUENCE
{
  x-position INTEGER,
  y-position INTEGER
}

END
```

ANNEXE B

Notation textuelle pour les applications MHEG-5

La présente annexe décrit la notation textuelle des objets MHEG-5 conformes à l'ISO/CEI 13522-5. La représentation alternative est la notation ASN.1 définie dans l'Annexe A.

B.1 Définitions générales

B.1.1 Code

Le jeu de code à utiliser pour la notation textuelle sera le jeu ISO/CEI 646 "Technologies de l'information – Jeu ISO de caractères codés sur 7 bits pour l'échange d'information". La notation textuelle utilisera un sous-ensemble de l'ISO/CEI 646 qui sera l'intervalle de caractères allant de SP (0x20) à ~ (0x7e), plus HT (0x09), LF (0x0a), FF (0x0c), et CR (0x0d).

Les autres caractères ne seront pas utilisés

NOTE 1 – En dépit du fait que la notation textuelle limite les codes de caractères à utiliser, les contenus de données 8 bits peuvent être codés au moyen de QPRINTABLE (voir B.3.4), BASE64 (voir B.3.5) et les contenus externes référencés par *ContentReference*.

NOTE 2 – Un domaine applicatif pourra étendre de jeu de caractères à utiliser dans la notation textuelle tant qu'il ne transgresse pas la grammaire. Par exemple, les caractères allant de 0x80 à 0xfe pourraient aussi être autorisés pour les chaînes et les commentaires.

B.1.2 Délimiteur

HT (0x09), LF (0x0a), FF (0x0c), CR (0x0d) et SP (0x20) sont appelés des délimiteurs.

La grammaire décrite dans la notation textuelle est basée sur les mots. Un mot est soit une parenthèse ("(" ou ")"), une accolade ("{" ou "}"), une balise (voir B.1.4) ou un symbole terminal (voir B.3). N'importe quel nombre de délimiteurs peut être inséré entre deux mots adjacents, sans en changer l'interprétation des mots. Cependant, au moins un délimiteur existera entre deux symboles terminaux et entre toute balise et tout symbole terminal, puisque autrement, ils seraient interprétés comme un simple terminal ou une simple balise.

B.1.3 Commentaire

"//" (0x2f 0x2f) n'étant pas dans un STRING, QPRINTABLE et BASE64 (voir B.3.3, B.3.4 et B.3.5) est utilisé pour indiquer le début d'un commentaire. Tous les caractères entre "//" (incluant le //) et l'apparition suivante de LF (0x0a), FF (0x0c) ou CR (0x0d) seront ignorés.

NOTE – HT (0x09) et SP (0x20) n'indiquent jamais la fin d'un commentaire.

B.1.4 Balise

Un jeton commençant par ":" (0x3a) est appelé "balise". Une balise est précédée de "{" (0x7b) quand il est utilisé au début d'un objet MHEG-5. Une balise est utilisée pour distinguer les objets MHEG-5 et leurs valeurs d'attributs associées. Les tags ne sont pas tiennent pas compte des différences entre majuscules et minuscules, à savoir ":Root", ":root", ":ROOT", ":rOOt" etc. sont tous identiques. Cependant , dans cette notation textuelle, des combinaisons de majuscules et de minuscules dans les balises sont utilisées pour en faciliter la lecture et la compréhension.

B.2 Définitions des symboles

Le Tableau B.1 suivant montre les symboles utilisés dans la notation textuelle ainsi que leurs significations respectives.

Tableau B.1/T.172 – Définitions des symboles dans la notation textuelle

Symbole	Définition
::=	est défini être
|	alternative
<">	double quote (0x22)
"text"	littéral contenu entre double quotes
<text>	description textuelle expliquant les codes ici présents
*	l'unité syntaxique précédente peut être répétée zéro ou plusieurs fois
+	l'unité syntaxique précédente peut être répétée une ou plusieurs fois
[]	l'unité syntaxique incluse est optionnelle. Elle peut apparaître zéro ou plusieurs fois
.	fin de paragraphe

B.3 Symboles terminaux

Tous les symboles terminaux utilisés dans la notation textuelle sont définis comme suit.

B.3.1 Entier (INTEGER)

Une valeur entière décimale ou hexadécimale positive.

Définition:

```
INTEGER      ::= DECINT | HEXINT | "0".
DECINT       ::= ["-"] DIGIT [DIGIT0]*.
DIGIT        ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
DIGIT0       ::= DIGIT | "0".
HEXINT       ::= HEXPREFIX HEXDIGIT0 [HEXDIGIT0]*.
HEXPREFIX    ::= "0x" | "0X".
HEXDIGIT0    ::= DIGIT | "0" | "a" | "b" | "c" | "d" | "e" | "f" | "A" | "B" | "C" | "D" | "E" | "F".
```

B.3.2 Booleen (BOOLEAN)

Une valeur booléenne peut être soit vraie ou fausse. Le terminal BOOLEAN n'est pas sensible à la différence entre majuscules et minuscules. "True", "TRUE" et "true" sont identiques, "False", "FALSE" et "false" sont identiques.

Définition:

```
BOOLEAN      ::= "true" | "false".
```

B.3.3 Chaîne (STRING)

Une valeur de chaîne incluse entre guillemets peut contenir un nombre arbitraire des caractères imprimables (de 0x20 à 0x7e). Le guillemet (0x22) à l'intérieur d'une STRING sera codée comme " \" " (0x5c 0x22) et barre oblique inverse (0x5c) sera codée "\\" (0x5c 0x5c).

Il faut noter qu'aucun retour à la ligne n'est inclus dans une STRING, à savoir les contenus tenant sur plusieurs lignes seront imprimés en utilisant QPRINTABLE ou BASE64.

Définition:

```
STRING       ::= <"> STRINGCHAR* <">.
STRINGCHAR   ::= <tout caractère imprimable simple sauf <"> et "\"> | "\"" | "\\".
```

B.3.4 Chaîne (QPRINTABLE)

Une valeur de chaîne incluse entre des guillemets simples contiendra un contenu codé de type imprimable comme défini dans RFC 1521. Cependant, " ' " (0x27) sera codé comme =27. Les minuscules tels que "a", "b", "c", "d", "e" et "f" peuvent être utilisées comme une représentation générale sur 8 bits définie par la section 5.1 règle #1 dans RFC 1521. Le nombre de caractères dans une ligne n'est pas limité. Les sauts de ligne ne seront pas convertis en CR (0x0d)/LF (0x0a); cependant au moins l'un des caractères suivants : LF (0x0a), FF (0x0c) and CR (0x0d) sera représenté au moyen d'un saut de ligne.

Définition:

QPRINTABLE ::= "'" QPRINTABLECHAR* "'" .
QPRINTABLECHAR ::= <un caractère tel que défini ci-dessus > | <séquence de code d'un caractère comme définie ci-dessus > .

B.3.5 (Chaîne) BASE64

Une valeur de chaîne incluse entre des guillemets inversés contiendra un contenu codé BASE64 comme défini dans RFC 1521. Cependant, le nombre de caractères dans une ligne n'est pas limité dans ce standard. Le contenu codé BASE64 peut être découpé en plusieurs segments par au moins un LF (0x0a), FF (0x0c) ou CR (0x0d). Cependant, ces caractères seront ignorés et les segments en entrée codés BASE64 seront manipulés comme s'ils formaient une seule ligne.

Définition:

BASE64 ::= "`" BASE64CHAR* "`" .
BASE64CHAR ::= <une séquence de code d'un caractère comme définie ci-dessus > .

B.3.6 Null

Null représente un terminal spécial dont la sémantique dépend de la définition des objets MHEG-5. Le terminal Null n'est pas sensible à la différence entre majuscules ou minuscules, à savoir "NULL" et "null" sont identiques.

Définition:

Null ::= "NULL".

B.3.7 Valeurs énumérées

Un mot terminal commençant avec une lettre de l'alphabet est appelé " valeur énumérée " (toutes les valeurs énumérées sont incluses entre deux doubles guillemets dans la grammaire qui suit). Une valeur énumérée est utilisée comme un des symboles terminaux représentants une sémantique spécifique dépendant de l'usage qui en est fait. Les énumérations ne sont pas sensibles aux différences entre majuscules ou minuscules, à savoir "*IsAvailable*" et "*isavailable*" sont identiques.

B.4 Définition d'objets MHEG-5

La notation textuelle des objets MHEG-5 est définie comme suit.

Le Tableau B.2 résume les abréviations utilisées dans les balises qui leur donnent une longueur raisonnable tout en préservant leur caractère lisible.

Tableau B.2/T.172 – Abréviations de balises

Bordered Bounding Box	BBBox
Content Cache Priority	CCPriority
Content Hook	Chook
Coordinate System	CS
Generic	G
Group Cache Priority	GCPriority
Horizontal Justification	HJustification
Identifier	ID
Information	Info
Interchanged	Interchg
OctetString	Ostring
Original	Orig
Program	Prg
Reference	Ref
Register	Reg
Standard	Std
Variable	Var (sauf pour les étiquettes d'action élémentaire)
Vertical Justification	VJustification

B.4.1 classe Root

```
Root            ::= ObjectIdentifier .
ObjectIdentifier    ::= ObjectReference .
```

B.4.2 classe Group

```
Group           ::= Root [StandardIdentifier]
                    [StandardVersion] [ObjectInformation]
                    [OnStartUp] [OnCloseDown]
                    [OriginalGroupCachePriority] [Items].
StandardIdentifier          ::= ":StdID" JointIsoItuIdentifier
                    MHEGStandardIdentifier.
JointIsoItuIdentifier       ::= INTEGER.
MHEGStandardIdentifier          ::= INTEGER.
StandardVersion             ::= ":StdVersion" INTEGER.
ObjectInformation           ::= ":ObjectInfo" OctetString.
OnStartUp           ::= ":OnStartUp" ActionClass.
OnCloseDown                 ::= ":OnCloseDown" ActionClass.
OriginalGroupCachePriority              ::= ":OrigGCPriority" INTEGER.
Items           ::= ":Items" "(" GroupItem+ ")".
GroupItem           ::= ResidentProgramClass |
                    RemoteProgramClass |
                    InterchangedProgramClass |
                    PaletteClass |
                    FontClass |
                    CursorShapeClass |
                    BooleanVariableClass |
                    IntegerVariableClass |
```

OctetStringVariableClass |
ObjectRefVariableClass |
ContentRefVariableClass |
LinkClass |
StreamClass |
BitmapClass |
LineArtClass |
DynamicLineArtClass |
RectangleClass |
HotspotClass |
SwitchButtonClass |
PushButtonClass |
TextClass |
EntryFieldClass |
HyperTextClass |
SliderClass |
TokenGroupClass |
ListGroupClass.

B.4.3 classe Application

```
ApplicationClass    ::= "{:Application" Group
                [OnSpawnCloseDown] [OnRestart]
                [DefaultAttributes] "}".
OnSpawnCloseDown        ::= ":OnSpawnCloseDown" ActionClass.
OnRestart    ::= ":OnRestart" ActionClass.
DefaultAttributes    ::= DefaultAttribute+.
DefaultAttribute    ::= CharacterSet | BackgroundColour
                | TextContentHook
                | TextColour | Font
                | FontAttributes
                | InterchangedProgramContentHook
                | StreamContentHook
                | BitmapContentHook
                | LineArtContentHook | ButtonRefColour
                | HighlightRefColour | SliderRefColour.
CharacterSet        ::= ":CharacterSet" INTEGER.
BackgroundColour        ::= ":BackgroundColour" Colour.
TextContentHook                 ::= ":TextCHook" INTEGER.
TextColour          ::= ":TextColour" Colour.
Font            ::= ":Font" FontBody.
FontBody    ::= DirectFont | IndirectFont.
DirectFont          ::= OctetString.
IndirectFont        ::= ObjectReference.
FontAttributes              ::= ":FontAttributes" OctetString.
InterchangedProgramContentHook      ::= ":InterchgPrgCHook" INTEGER.
StreamContentHook       ::= ":StreamCHook" INTEGER.
BitmapContentHook       ::= ":BitmapCHook" INTEGER.
LineArtContentHook              ::= ":LineArtCHook" INTEGER.
ButtonRefColour         ::= ":ButtonRefColour" Colour.
HighlightRefColour      ::= ":HighlightRefColour" Colour.
SliderRefColour         ::= ":SliderRefColour" Colour.
```

B.4.4 classe Scene

```
SceneClass          ::= "{:Scene" Group InputEventRegister
                SceneCoordinateSystem [AspectRatio]
                [MovingCursor] [NextScenes] "}".
InputEventRegister      ::= ":InputEventReg" INTEGER.
SceneCoordinateSystem           ::= ":SceneCS" XScene YScene.
```

```
XScene              ::= INTEGER.
YScene              ::= INTEGER.
AspectRatio         ::= ":AspectRatio" Width Height.
Width       ::= INTEGER.
Height      ::= INTEGER.
MovingCursor            ::= ":MovingCursor" BOOLEAN.
NextScenes          ::= ":NextScenes" "(" NextScene+ ")".
NextScene           ::= "(" SceneRef SceneWeight ")".
SceneRef    ::= OctetString.
SceneWeight         ::= INTEGER.
```

B.4.5 classe Ingredient

```
Ingredient  ::= Root [InitiallyActive] [ContentHook]
            [OriginalContent] [Shared].
InitiallyActive             ::= ":InitiallyActive" BOOLEAN.
ContentHook         ::= ":CHook" INTEGER.
OriginalContent             ::= ":OrigContent" ContentBody.
ContentBody         ::= IncludedContent | ReferencedContent.
IncludedContent             ::= OctetString.
ReferencedContent           ::= ":ContentRef" "(" ContentReference
            [ContentSize] [ContentCachePriority]
            ")".
ContentSize         ::= ":ContentSize" INTEGER.
ContentCachePriority            ::= ":CCPriority" INTEGER.
Shared              ::= ":Shared" BOOLEAN.
```

B.4.6 classe Link

```
LinkClass   ::= "{:Link" Ingredient LinkCondition
            LinkEffect "}".
LinkCondition               ::= EventSource EventType [EventData].
EventSource         ::= ":EventSource" ObjectReference.
EventType           ::= ":EventType" EventTypeEnum.
EventTypeEnum               ::= "IsAvailable" | "ContentAvailable"
            | "IsDeleted" | "IsRunning"
            | "IsStopped" | "UserInput"
            | "AnchorFired" | "TimerFired"
            | "AsynchStopped" | "InteractionCompleted"
            | "TokenMovedFrom" | "TokenMovedTo"
            | "StreamEvent" | "StreamPlaying"
            | "StreamStopped" | "CounterTrigger"
            | "HighlightOn" | "HighlightOff"
            | "CursorEnter" | "CursorLeave"
            | "IsSelected" | "IsDeselected"
            | "TestEvent" | "FirstItemPresented"
            | "LastItemPresented" | "HeadItems"
            | "TailItems" | "ItemSelected"
            | "ItemDeselected" | "EntryFieldFull"
                                | "EngineEvent".
EventData           ::= ":EventData" EventDataBody.
EventDataBody               ::= OctetString | BOOLEAN | INTEGER.
LinkEffect          ::= ":LinkEffect" ActionClass.
```

B.4.7 classe Program

```
Program             ::= Ingredient Name [InitiallyAvailable].
Name        ::= ":Name" OctetString.
InitiallyAvailable  ::= ":InitiallyAvailable" BOOLEAN.
```

B.4.8 classe ResidentProgram

```
ResidentProgramClass     ::= "{:ResidentPrg" Program "}".
```

B.4.9 classe RemoteProgram

```
RemoteProgramClass       ::= "{:RemotePrg" Program
                [ProgramConnectionTag] "}".
ProgramConnectionTag     ::= ":ConnectionTag" INTEGER.
```

B.4.10 classe InterchangedProgram

```
InterchangedProgramClass        ::= "{:InterchgPrg" Program "}".
```

B.4.11 classe Palette

```
PaletteClass          ::= "{:Palette" Ingredient "}".
```

B.4.12 classe Font

```
FontClass     ::= "{:Font" Ingredient "}".
```

B.4.13 classe CursorShape

```
CursorShapeClass  ::= "{:CursorShape" Ingredient "}".
```

B.4.14 classe Variable

```
Variable       ::= Ingredient OriginalValue.
OriginalValue        ::= ":OrigValue" OriginalValueBody.
OriginalValueBody          ::= BOOLEAN | INTEGER | OctetString
                | ObjectReferenceValue
                | ContentReferenceValue.
ObjectReferenceValue      ::= ":ObjectRef" ObjectReference.
ContentReferenceValue     ::= ":ContentRef" ContentReference.
```

B.4.15 classe BooleanVariable

```
BooleanVariableClass      ::= "{:BooleanVar" Variable "}".
```

B.4.16 classe IntegerVariable

```
IntegerVariableClass      ::= "{:IntegerVar" Variable "}".
```

B.4.17 classe OctetStringVariable

```
OctetStringVariableClass          ::= "{:OStringVar" Variable "}".
```

B.4.18 classe ObjectRefVariable

```
ObjectRefVariableClass    ::= "{:ObjectRefVar" Variable "}".
```

B.4.19 classe ContentRefVariable

```
ContentRefVariableClass          ::= "{:ContentRefVar" Variable "}".
```

B.4.20 classe Presentable

```
Presentable           ::= Ingredient.
```

B.4.21 classe TokenManager

```
TokenManager            ::= [MovementTable].
MovementTable           ::= ":MovementTable" "(" Movement+ ")".
Movement          ::= "(" TargetElement+ ")".
TargetElement           ::= INTEGER.
```

B.4.22 classe TokenGroup

```
TokenGroupClass         ::= "{:TokenGroup" TokenGroupBody "}".
TokenGroupBody          ::= Presentable TokenManager TokenGroupItems
            [NoTokenActionSlots].
TokenGroupItems         ::= ":TokenGroupItems" "(" TokenGroupItem+ ")".
TokenGroupItem          ::= "(" AVisible [ActionSlots] ")".
AVisible      ::= ObjectReference.
ActionSlots             ::= ":ActionSlots" "(" ActionSlot+ ")".
ActionSlot              ::= ActionClass | Null.
NoTokenActionSlots              ::= ":NoTokenActionSlots" "(" ActionSlot+ ")".
```

B.4.23 classe ListGroup

```
ListGroupClass          ::= "{:ListGroup" TokenGroupBody
            Positions [WrapAround]
            [MultipleSelection] "}".
Positions     ::= ":Positions" "(" Position+ ")".
Position      ::= "(" XYPosition ")".
WrapAround          ::= ":WrapAround" BOOLEAN.
MultipleSelection       ::= ":MultipleSelection" BOOLEAN.
```

B.4.24 classe Visible

```
Visible       ::= Presentable OriginalBoxSize
      [OriginalPosition] [OriginalPaletteRef].
OriginalBoxSize     ::= ":OrigBoxSize" BoxSize.
BoxSize       ::= XLength YLength.
XLength       ::= INTEGER.
YLength       ::= INTEGER.
OriginalPosition    ::= ":OrigPosition" XYPosition.
OriginalPaletteRef      ::= ":OrigPaletteRef" ObjectReference.
```

B.4.25 classe Bitmap

```
BitmapClass         ::= "{:Bitmap" Visible [Tiling]
            [OriginalTransparency] "}".
Tiling          ::= ":Tiling" BOOLEAN.
OriginalTransparency      ::= ":OrigTransparency" INTEGER.
```

B.4.26 classe LineArt

```
LineArtClass        ::= "{:LineArt" LineArtBody "}".
LineArtBody         ::= Visible [BorderedBoundingBox]
            [OriginalLineWidth]
            [OriginalLineStyle]
            [OriginalRefLineColour]
            [OriginalRefFillColour] "}".
BorderedBoundingBox     ::= ":BBBox" BOOLEAN.
OriginalLineWidth ::= ":OrigLineWidth" INTEGER.
OriginalLineStyle   ::= ":OrigLineStyle" INTEGER.
OriginalRefLineColour     ::= ":OrigRefLineColour" Colour.
OriginalRefFillColour     ::= ":OrigRefFillColour" Colour.
```

B.4.27 classe Rectangle

```
RectangleClass      ::= "{:Rectangle" LineArtBody "}".
```

B.4.28 classe DynamicLineArt

```
DynamicLineArtClass     ::= "{:DynamicLineArt" LineArtBody "}".
```

B.4.29 classe Text

```
TextClass    ::= "{:Text" TextBody "}".
TextBody     ::= Visible [OriginalFont] [FontAttributes]
                 [TextColour] [BackgroundColour]
                 [CharacterSet]
                 [HorizontalJustification]
                 [VerticalJustification]
                 [LineOrientation] [StartCorner]
                 [TextWrapping].
OriginalFont::= ":OrigFont" FontBody.
HorizontalJustification          ::= ":HJustification" JustificationEnum.
JustificationEnum          ::= "start" | "end" | "centre" | "justified".
VerticalJustification      ::= ":VJustification" JustificationEnum.
LineOrientation            ::= ":LineOrientation" LineOrientationEnum.
LineOrientationEnum              ::= "vertical" | "horizontal".
StartCorner          ::= ":StartCorner" StartCornerEnum.
StartCornerEnum                  ::= "upper-left" | "upper-right"
                 | "lower-left" | "lower-right".
TextWrapping                     ::= ":TextWrapping" BOOLEAN.
```

B.4.30 classe Stream

```
StreamClass          ::= "{:Stream" Presentable Multiplex
                 [Storage] [Looping] "}".
Multiplex    ::= ":Multiplex" "(" StreamComponent+ ")".
StreamComponent          ::= AudioClass | VideoClass | RTGraphicsClass.
Storage              ::= ":Storage" StorageEnum.
StorageEnum          ::= "memory" | "stream".
Looping              ::= ":Looping" INTEGER.
```

B.4.31 classe Audio

```
AudioClass           ::= "{:Audio" Presentable ComponentTag
                 [OriginalVolume] "}".
ComponentTag             ::= ":ComponentTag" INTEGER.
OriginalVolume           ::= ":OrigVolume" INTEGER.
```

B.4.32 classe Video

```
VideoClass           ::= "{:Video" Visible ComponentTag
                 [Termination].
Termination          ::= ":Termination" TerminationEnum.
TerminationEnum                  ::= "freeze" | "disappear".
```

B.4.33 classe RTGraphics

```
RTGraphicsClass                  ::= "{:RTGraphics" Visible ComponentTag
                 [Termination] "}".
```

B.4.34 classe Interactible

```
Interactible            ::= [EngineResp] [HighlightRefColour].
EngineResp              ::= ":EngineResp" BOOLEAN.
```

B.4.35 classe Slider

```
SliderClass             ::= "{:Slider" Visible Interactible
            Orientation MaxValue [MinValue]
            [InitialValue] [InitialPortion]
            [StepSize] [SliderStyle]
            [SliderRefColour] "}".
Orientation         ::= ":Orientation" OrientationEnum.
OrientationEnum         ::= "left" | "right" | "up" | "down".
MaxValue            ::= ":MaxValue" INTEGER.
MinValue    ::= ":MinValue" INTEGER.
InitialValue        ::= ":InitialValue" INTEGER.
InitialPortion      ::= ":InitialPortion" INTEGER.
StepSize    ::= ":StepSize" INTEGER.
SliderStyle         ::= ":SliderStyle" SliderStyleEnum.
SliderStyleEnum         ::= "normal" | "thermometer" | "proportional".
```

B.4.36 classe EntryField

```
EntryFieldClass         ::= "{:EntryField" TextBody Interactible
            [InputType] [CharList]
            [ObscuredInput] [MaxLength] "}".
InputType   ::= ":InputType" InputTypeEnum.
InputTypeEnum   ::= "alpha" | "numeric" | "any" | "listed".
CharList    ::= ":CharList" OctetString.
ObscuredInput   ::= ":ObscuredInput" BOOLEAN.
MaxLength   ::= ":MaxLength" INTEGER.
```

B.4.37 classe HyperText

```
HyperTextClass      ::= "{:HyperText" TextBody Interactible
            "}".
```

B.4.38 classe Button

```
Button      ::= Visible Interactible [ButtonRefColour].
```

B.4.39 classe Hotspot

```
HotspotClass        ::= "{:Hotspot" Button "}".
```

B.4.40 classe PushButton

```
PushButtonClass     ::= "{:PushButton" PushButtonBody "}".
PushButtonBody      ::= Button [OriginalLabel] [CharacterSet].
OriginalLabel           ::= ":OrigLabel" OctetString.
```

B.4.41 classe SwitchButton

```
SwitchButtonClass ::= "{:SwitchButton" PushButtonBody
            ButtonStyle "}".
ButtonStyle  ::= ":ButtonStyle" ButtonStyleEnum.
ButtonStyleEnum         ::= "pushbutton" | "radiobutton" | "checkbox".
```

B.4.42 classe Action

ActionClass ::= "(" ElementaryAction+ ")".
ElementaryAction ::= Activate
| Add
| AddItem
| Append
| BringToFront
| Call
| CallActionSlot
| Clear
| Clone
| CloseConnection
| Deactivate
| DelItem
| Deselect
| DeselectItem
| Divide
| DrawArc
| DrawLine
| DrawOval
| DrawPolygon
| DrawPolyline
| DrawRectangle
| DrawSector
| Fork
| GetAvailabilityStatus
| GetBoxSize
| GetCellItem
| GetCursorPosition
| GetEngineSupport
| GetEntryPoint
| GetFillColour
| GetFirstItem
| GetHighlightStatus
| GetInteractionStatus
| GetItemStatus
| GetLabel
| GetLastAnchorFired
| GetLineColour
| GetLineStyle
| GetLineWidth
| GetListItem
| GetListSize
| GetOverwriteMode
| GetPortion
| GetPosition
| GetRunningStatus
| GetSelectionStatus
| GetSliderValue
| GetTextContent
| GetTextData
| GetTokenPosition
| GetVolume
| Launch
| LockScreen
| Modulo
| Move
| MoveTo
| Multiply

| OpenConnection
| Preload
| PutBefore
| PutBehind
| Quit
| ReadPersistent
| Run
| ScaleBitmap
| ScaleVideo
| ScrollItems
| Select
| SelectItem
| SendEvent
| SendToBack
| SetBoxSize
| SetCachePriority
| SetCounterEndPosition
| SetCounterPosition
| SetCounterTrigger
| SetCursorPosition
| SetCursorShape
| SetData
| SetEntryPoint
| SetFillColour
| SetFirstItem
| SetFontRef
| SetHighlightStatus
| SetInteractionStatus
| SetLabel
| SetLineColour
| SetLineStyle
| SetLineWidth
| SetOverwriteMode
| SetPaletteRef
| SetPortion
| SetPosition
| SetSliderValue
| SetSpeed
| SetTimer
| SetTransparency
| SetVariable
| SetVolume
| Spawn
| Step
| Stop
| StorePersistent
| Subtract
| TestVariable
| Toggle
| ToggleItem
| TransitionTo
| Unload
| UnlockScreen.

Activate ::= ":Activate" "(" Target ")".
Add ::= ":Add" "(" Target Value ")".
AddItem ::= ":AddItem" "(" Target ItemIndex VisibleReference ")".
Append ::= ":Append" "(" Target AppendValue ")".
BringToFront ::= ":BringToFront" "(" Target ")".

```
Call            ::= ":Call" "(" Target CallSucceeded
                    [Parameters] ")".
CallActionSlot      ::= ":CallActionSlot" "(" Target Index ")".
Clear           ::= ":Clear" "(" Target ")".
Clone           ::= ":Clone" "(" Target CloneRefVar ")".
CloseConnection     ::= ":CloseConnection" "(" Target
                    ConnectionTag ")".
Deactivate          ::= ":Deactivate" "(" Target ")".
DelItem             ::= ":DelItem" "(" Target VisibleReference
                    ")".
Deselect        ::= ":Deselect" "(" Target ")".
DeselectItem        ::= ":DeselectItem" "(" Target ItemIndex ")".
Divide          ::= ":Divide" "(" Target Value ")".
DrawArc             ::= ":DrawArc" "(" Target X Y EllipseWidth
                    EllipseHeight StartAngle ArcAngle ")".
DrawLine        ::= ":DrawLine" "(" Target X1 Y1 X2 Y2 ")".
DrawOval        ::= ":DrawOval" "(" Target X Y EllipseWidth
                    EllipseHeight ")".
DrawPolygon         ::= ":DrawPolygon" "(" Target PointList ")".
DrawPolyline        ::= ":DrawPolyline" "(" Target PointList ")".
DrawRectangle           ::= ":DrawRectangle" "(" Target X1 Y1 X2 Y2
                    ")".
DrawSector          ::= ":DrawSector" "(" Target X Y EllipseWidth
                    EllipseHeight StartAngle ArcAngle ")".
Fork            ::= ":Fork" "(" Target ForkSucceeded
                    [Parameters] ")".
GetAvailabilityStatus       ::= ":GetAvailabilityStatus" "(" Target
                    AvailabilityStatusVar ")".
GetBoxSize          ::= ":GetBoxSize" "(" Target XBoxSizeVar
                    YBoxSizeVar ")".
GetCellItem         ::= ":GetCellItem" "(" Target CellIndex
                    ItemRefVar ")".
GetCursorPosition           ::= ":GetCursorPosition" "(" Target XOut YOut
                    ")".
GetEngineSupport            ::= ":GetEngineSupport" "(" Target Feature
                    Answer ")".
GetEntryPoint               ::= ":GetEntryPoint" "(" Target EntryPointVar
                    ")".
GetFillColour               ::= ":GetFillColour" "(" Target FillColourVar
                    ")".
GetFirstItem        ::= ":GetFirstItem" "(" Target FirstItemVar
                    ")".
GetHighlightStatus          ::= ":GetHighlightStatus" "(" Target
                    HighlightStatusVar ")".
GetInteractionStatus        ::= ":GetInteractionStatus" "(" Target
                    InteractionStatusVar ")".
GetItemStatus               ::= ":GetItemStatus" "(" Target
                    ItemIndex ItemStatusVar ")".
GetLabel        ::= ":GetLabel" "(" Target LabelVar ")".
GetLastAnchorFired          ::= ":GetLastAnchorFired" "(" Target
                    LastAnchorFiredVar ")".
GetLineColour               ::= ":GetLineColour" "(" Target LineColourVar
                    ")".
GetLineStyle        ::= ":GetLineStyle" "(" Target LineStyleVar
                    ")".
GetLineWidth                ::= ":GetLineWidth" "(" Target LineWidthVar
                    ")".
GetListItem         ::= ":GetListItem" "(" Target ItemIndex
                    ItemRefVar ")".
GetListSize         ::= ":GetListSize" "(" Target SizeVar ")".
```

```
GetOverwriteMode        ::= ":GetOverwriteMode" "(" Target
                OverwriteModeVar ")".
GetPortion              ::= ":GetPortion" "(" Target PortionVar ")".
GetPosition             ::= ":GetPosition" "(" Target XPositionVar
                YPositionVar ")".
GetRunningStatus        ::= ":GetRunningStatus" "(" Target
                RunningStatusVar ")".
GetSelectionStatus      ::= ":GetSelectionStatus" "(" Target
                SelectionStatusVar ")".
GetSliderValue          ::= ":GetSliderValue" "(" Target
                SliderValueVar ")".
GetTextContent          ::= ":GetTextContent" "(" Target
                TextContentVar ")".
GetTextData             ::= ":GetTextData" "(" Target TextDataVar ")".
GetTokenPosition            ::= ":GetTokenPosition" "(" Target
                TokenPositionVar ")".
GetVolume               ::= ":GetVolume" "(" Target VolumeVar ")".
Launch                  ::= ":Launch" "(" Target ")".
LockScreen              ::= ":LockScreen" "(" Target ")".
Modulo                  ::= ":Modulo" "(" Target Value ")".
Move        ::= ":Move" "(" Target MovementIdentifier ")".
MoveTo                  ::= ":MoveTo" "(" Target Index ")".
Multiply    ::= ":Multiply" "(" Target Value ")".
OpenConnection          ::= ":OpenConnection" "(" Target OpenSucceeded
                Protocol Address ConnectionTag ")".
Preload             ::= ":Preload" "(" Target ")".
PutBefore   ::= ":PutBefore" "(" Target ReferenceVisible
                ")".
PutBehind           ::= ":PutBehind" "(" Target ReferenceVisible
                ")".
Quit        ::= ":Quit" "(" Target ")".
ReadPersistent          ::= ":ReadPersistent" "(" Target ReadSucceeded
                OutVariables InFileName ")".
Run         ::= ":Run" "(" Target ")".
ScaleBitmap         ::= ":ScaleBitmap" "(" Target XScale YScale
                ")".
ScaleVideo          ::= ":ScaleVideo" "(" Target XScale YScale ")".
ScrollItems         ::= ":ScrollItems" "(" Target ItemsToScroll ")".
Select      ::= ":Select" "(" Target ")".
SelectItem          ::= ":SelectItem" "(" Target ItemIndex ")".
SendEvent           ::= ":SendEvent" "(" Target EmulatedEventSource
                EmulatedEventType [EmulatedEventData]
                ")".
SendToBack          ::= ":SendToBack" "(" Target ")".
SetBoxSize          ::= ":SetBoxSize" "(" Target XNewBoxSize
                YNewBoxSize ")" .
SetCachePriority    ::= ":SetCachePriority" "(" Target
                NewCachePriority ")".
SetCounterEndPosition           ::= ":SetCounterEndPosition" "(" Target
                NewCounterEndPosition ")".
SetCounterPosition      ::= ":SetCounterPosition" "(" Target
                NewCounterPosition ")".
SetCounterTrigger       ::= ":SetCounterTrigger" "(" Target
                TriggerIdentifier [NewCounterValue]
                ")".
SetCursorPosition       ::= ":SetCursorPosition" "(" Target XCursor
                YCursor ")".
SetCursorShape          ::= ":SetCursorShape" "(" Target
                [NewCursorShape] ")".
SetData             ::= ":SetData" "(" Target NewContent ")".
```

```
SetEntryPoint           ::= ":SetEntryPoint" "(" Target NewEntryPoint
                ")".
SetFillColour       ::= ":SetFillColour" "(" Target [NewColour]
                ")".
SetFirstItem        ::= ":SetFirstItem" "(" Target NewFirstItem
                ")".
SetFontRef          ::= ":SetFontRef" "(" Target NewFont ")".
SetHighlightStatus          ::= ":SetHighlightStatus" "(" Target
                NewHighlightStatus ")".
SetInteractionStatus        ::= ":SetInteractionStatus" "(" Target
                NewInteractionStatus ")".
SetLabel    ::= ":SetLabel" "(" Target NewLabel ")".
SetLineColour           ::= ":SetLineColour" "(" Target NewColour ")".
SetLineStyle        ::= ":SetLineStyle" "(" Target NewLineStyle
                ")".
SetLineWidth            ::= ":SetLineWidth" "(" Target NewLineWidth
                ")".
SetOverwriteMode        ::= ":SetOverwriteMode" "(" Target
                NewOverwriteMode ")".
SetPaletteRef       ::= ":SetPaletteRef" "(" Target NewPaletteRef
                ")".
SetPortion          ::= ":SetPortion" "(" Target NewPortion ")".
SetPosition         ::= ":SetPosition" "(" Target NewXPosition
                NewYPosition ")".
SetSliderValue          ::= ":SetSliderValue" "(" Target
                NewSliderValue ")".
SetSpeed    ::= ":SetSpeed" "(" Target NewSpeed ")".
SetTimer    ::= ":SetTimer" "(" Target TimerID
                [TimerValue] [AbsoluteTime] ")".
SetTransparency         ::= ":SetTransparency" "(" Target
                NewTransparency ")".
SetVariable         ::= ":SetVariable" "(" Target
                NewVariableValue ")".
SetVolume           ::= ":SetVolume" "(" Target NewVolume ")".
Spawn       ::= ":Spawn" "(" Target ")".
Stop        ::= ":Stop" "(" Target ")".
Step        ::= ":Step" "(" Target NbOfSteps ")".
StorePersistent         ::= ":StorePersistent" "(" Target
                StoreSucceeded InVariables OutFileName
                ")".
Subtract    ::= ":Subtract" "(" Target Value ")".
TestVariable        ::= ":TestVariable" "(" Target Operator
                ComparisonValue ")".
Toggle          ::= ":Toggle" "(" Target ")".
ToggleItem          ::= ":ToggleItem" "(" Target ItemIndex ")".
TransitionTo::= ":TransitionTo" "(" Target [ConnectionTag]
                [TransitionEffect] ")".
Unload          ::= ":Unload" "(" Target ")".
UnlockScreen            ::= ":UnlockScreen" "(" Target ")".

AbsoluteTime        ::= ":AbsoluteTime" GenericBoolean.
Address         ::= GenericOctetString.
Answer          ::= ObjectReference.
AppendValue         ::= GenericOctetString.
ArcAngle    ::= GenericInteger.
AvailabilityStatusVar           ::= ObjectReference.
CallSucceeded           ::= ObjectReference.
CellIndex   ::= GenericInteger.
CloneRefVar         ::= ObjectReference.
```

ComparisonValue ::= NewGenericBoolean | NewGenericInteger
| NewGenericOctetString
| NewGenericObjectReference
| NewGenericContentReference.
ConnectionTag ::= ":ConnectionTag" GenericInteger.
Denominator::= GenericInteger.
EllipseHeight ::= GenericInteger.
EllipseWidth ::= GenericInteger.
EmulatedEventData ::= NewGenericBoolean | NewGenericInteger
| NewGenericOctetString.
EmulatedEventSource ::= GenericObjectReference .
EmulatedEventType ::= EventTypeEnum.
EntryPointVar ::= ObjectReference.
ForkSucceeded ::= ObjectReference.
Feature ::= GenericOctetString.
FillColourVar ::= ObjectReference.
FirstItemVar ::= ObjectReference.
HighlightStatusVar ::= ObjectReference.
Index ::= GenericInteger.
InFileName ::= GenericOctetString.
InteractionStatusVar ::= ObjectReference.
InVariables ::= "(" ObjectReference+ ")".
ItemIndex ::= GenericInteger.
ItemRefVar ::= ObjectReference.
ItemStatusVar ::= ObjectReference.
ItemsToScroll ::= GenericInteger.
LabelVar ::= ObjectReference.
LastAnchorFiredVar ::= ObjectReference.
LineColourVar ::= ObjectReference.
LineStyleVar ::= ObjectReference.
LineWidthVar ::= ObjectReference.
MovementIdentifier ::= GenericInteger.
NbOfSteps ::= GenericInteger.
NewAbsoluteColour ::= ":NewAbsoluteColour" GenericOctetString.
NewCachePriority ::= GenericInteger.
NewColour ::= NewColourIndex | NewAbsoluteColour.
NewColourIndex ::= ":NewColourIndex" GenericInteger.
NewContent ::= NewIncludedContent | NewReferencedContent.
NewContentCachePriority ::= ":NewCCPriority" GenericInteger.
NewCounterEndPosition ::= GenericInteger.
NewCounterPosition ::= GenericInteger.
NewContentSize ::= ":NewContentSize" GenericInteger.
NewCounterValue ::= GenericInteger.
NewCursorShape ::= GenericObjectReference.
NewEntryPoint ::= GenericInteger.
NewFirstItem ::= GenericInteger.
NewFont ::= NewFontName | NewFontReference.
NewFontName ::= NewGenericOctetString.
NewFontReference ::= NewGenericObjectReference.
NewGenericBoolean ::= ":GBoolean" GenericBoolean.
NewGenericInteger ::= ":GInteger" GenericInteger.
NewGenericOctetString ::= ":GOctetString" GenericOctetString.
NewGenericObjectReference ::= ":GObjectRef" GenericObjectReference.
NewGenericContentReference ::= ":GContentRef" GenericContentReference.
NewHighlightStatus ::= GenericBoolean.
NewIncludedContent ::= GenericOctetString.
NewInteractionStatus ::= GenericBoolean.
NewLabel ::= GenericOctetString.
NewLineStyle ::= GenericInteger.
NewLineWidth ::= GenericInteger.
NewOverwriteMode ::= GenericBoolean.

```
NewPaletteRef               ::= GenericObjectReference.
NewPortion          ::= GenericInteger.
NewReferencedContent            ::= ":NewRefContent" "(" GenericContentReference
                [NewContentSize]
                [NewContentCachePriority] ")".
NewSliderValue              ::= GenericInteger.
NewSpeed            ::= Rational.
NewTransparency                 ::= GenericInteger.
NewVariableValue            ::= NewGenericInteger | NewGenericBoolean
                | NewGenericOctetString
                | NewGenericObjectReference
                | NewGenericContentReference.
NewVolume           ::= GenericInteger.
NewXPosition                ::= GenericInteger.
NewYPosition                ::= GenericInteger.
Numerator           ::= GenericInteger.
OpenSucceeded               ::= ObjectReference.
Operator        ::= GenericInteger.
OutFileName         ::= GenericOctetString.
OutVariables        ::= "(" ObjectReference+ ")".
OverwriteModeVar            ::= ObjectReference.
Parameter       ::= NewGenericBoolean | NewGenericInteger
                | NewGenericOctetString
                | NewGenericObjectReference
                | NewGenericContentReference.
Parameters          ::= Parameter+.
Point           ::= "(" X Y ")".
PointList       ::= "(" Point+ ")".
PortionVar          ::= ObjectReference.
Protocol        ::= GenericOctetString.
Rational        ::= Numerator [Denominator].
ReadSucceeded               ::= ObjectReference.
ReferenceVisible        ::= GenericObjectReference.
RunningStatusVar            ::= ObjectReference.
SelectionStatusVar          ::= ObjectReference.
SizeVar             ::= ObjectReference.
SliderValueVar              ::= ObjectReference.
StartAngle          ::= GenericInteger.
StoreSucceeded              ::= ObjectReference.
Target          ::= GenericObjectReference.
TextContentVar              ::= ObjectReference.
TextDataVar         ::= ObjectReference.
TimerID             ::= GenericInteger.
TimerValue          ::= GenericInteger.
TokenPositionVar            ::= ObjectReference.
TransitionEffect        ::= GenericInteger.
TriggerIdentifier       ::= GenericInteger.
Value           ::= GenericInteger.
VisibleReference        ::= GenericObjectReference.
VolumeVar           ::= ObjectReference.
X       ::= GenericInteger.
X1      ::= GenericInteger.
X2      ::= GenericInteger.
XBoxSizeVar         ::= ObjectReference.
XCursor             ::= GenericInteger.
XNewBoxSize                 ::= GenericInteger.
XOut            ::= ObjectReference.
XPositionVar        ::= ObjectReference.
XScale              ::= GenericInteger.
```

```
Y       ::= GenericInteger.
Y1      ::= GenericInteger.
Y2      ::= GenericInteger.
YBoxSizeVar         ::= ObjectReference.
YCursor             ::= GenericInteger.
YNewBoxSize             ::= GenericInteger.
YOut        ::= ObjectReference.
YPositionVar        ::= ObjectReference.
YScale              ::= GenericInteger.
```

B.4.43 Références à Objet, Contenu, Valeur, Couleur et position

```
ObjectReference             ::= ExternalReference | InternalReference.
ExternalReference           ::= "(" GroupIdentifier ObjectNumber ")".
InternalReference           ::= ObjectNumber.
GroupIdentifier             ::= OctetString.
ObjectNumber                ::= INTEGER.

ContentReference            ::= OctetString.

GenericObjectReference          ::= DirectReference | IndirectReference.
DirectReference             ::= ObjectReference.
IndirectReference           ::= ":IndirectRef" ObjectReference.

GenericContentReference         ::= ContentReference | IndirectReference.

GenericInteger              ::= INTEGER | IndirectReference.

GenericBoolean              ::= BOOLEAN | IndirectReference.

GenericOctetString          ::= OctetString | IndirectReference.

OctetString             ::= STRING | QPRINTABLE | BASE64.

Colour      ::= ColourIndex | AbsoluteColour.
ColourIndex         ::= INTEGER.
AbsoluteColour              ::= OctetString.

XYPosition                  ::= XPosition YPosition.
XPosition           ::= INTEGER.
YPosition           ::= INTEGER.
```

APPENDICE I

Démarrage d'un moteur MHEG-5

Un moteur MHEG-5 peut être dans deux états: *idle* ou actif. Un moteur MHEG-5 est à l'état *idle* lorsque aucun objet *Application* n'est actif. A l'opposé, dans un moteur MHEG-5 actif un objet *Application* et un seul est actif. Le moteur passe de l'état actif à l'état *idle* seulement lors de l'exécution de l'action *Quit* dont la sémantique est détaillée dans le paragraphe 10. Le moteur passe de l'état *idle* à l'état actif:

1) soit comme une étape dans l'exécution des action *Launch* ou *Spawn* dont la sémantique est décrite dans le paragraphe 10;

2) soit comme le résultat d'un moteur MHEG-5 à l'état *idle* obligé par une action extérieure d'activer un objet *Application*.

Ce dernier cas est l'objet du présent sous-paragraphe.

Vu de l'extérieur, un moteur MHEG-5 est typiquement une application fonctionnant sur un dispositif dans un environnement logiciel. Lorsque cette application est activée pour la première fois, elle doit obtenir de la part de l'extérieur l'endroit où le premier objet *Application* peut être trouvé. Cette information peut provenir , par exemple de l'utilisateur, de la mémoire ou du réseau et devrait donner la possibilité au moteur MHEG-5 de se lier à l'espace de nom de l'objet *Application* et ainsi de trouver l'objet *Application* lui-même.

Une fois que le moteur MHEG-5 possède l'information, il essaie de récupérer cet objet *Application* et ensuite de l'activer. Ceci est fait en invoquant de manière implicite le comportement *Activation* de l'objet. Pour pouvoir aussi activer un objet *Scene* (sans lequel aucune saisie d'utilisateur ou d'affichage ne serait possible), l'objet *Application* devra posséder une action *TransitionTo* codée dans son objet *OnStartUp*; cette action activant ensuite le premier objet *Scene* de l'application.

APPENDICE II

Définition de domaines applicatifs

Le présent appendice spécifie les attributs de la présente Recommandation ayant besoin d'être définis par un domaine applicatif spécifique. Le but du présent appendice est de fournir un jeu d'outils pour créer une instance de moteur MHEG-5 et des exemples de valeurs de tables.

II.1 Format d'échange d'objet

On spécifiera par exemple ici que le domaine applicatif choisit comme format d'échange la notation ASN.1 définie dans l'Annexe A.

Interchange format: ASN.1.

II.2 Ensemble de classes

Le domaine applicatif choisit un ensemble de classes obligatoires telles que:

Action, Application, Audio, Bitmap, BooleanVariable, ContentRefVariable, EntryField, HotSpot, HyperText, IntegerVariable, InterchangedProgram, Link, ListGroup, ObjectRefVariable, OctetStringVariable, PushButton, Rectangle, RemoteProgram, ResidentProgram, Scene, Slider, Stream, SwitchButton, Text, TokenGroup, Video.

II.3 Ensemble d'attributs

Le domaine applicatif définit un ensemble d'attributs obligatoires et optionnels comme dans l'exemple suivant:

Attribut	Besoin
Ancillary connections	obligatoire
Caching	optionnel
Cloning	obligatoire
Free moving cursor	optionnel
Scaling (Video and Bitmap)	optionnel
Stacking of Applications	optionnel
Trick modes	optionnel

II.4 Codage de donnée de type contenu

La présente Recommandation est générique en ce sens qu'elle ne définit pas exactement comment les données de type contenu (comme une phototrame) sont codées. Cependant la présente Recommandation est spécifique en ce sens qu'elle:

1) spécifie quels types de données sont supportés;

2) fournit une méthode pour définir – Par domaine applicatif spécifique – une liste des codes réels utilisés dans ce domaine.

Le tableau suivant illustre comment un domaine applicatif pourrait spécifier les formats requis; les valeurs listées ci-dessous sont purement fournies à titre d'exemple de valeurs qu'un domaine applicatif pourrait choisir.

Table de contenu

Attribut	Valeurs permises
FontAttributes	*bold, italic, emphasis, strong emphasis*
FontName	(aucune spécification)
AbsoluteColour	RVB16, c'est-à-dire codage des graphiques en 16 bits par pixel, allouant 5 bits aux composantes Bleu et Rouge et 6 bits à la composante Vert
CharacterSet	1 – par référence au sous-ensemble de ISO/CEI 8859-1 comme spécifié dans HTML 2.0
TransitionEffect	(aucune spécification)

Table de codage

Type de contenu	Spécification (Types de données)	Valeurs de crochet
Font	(aucune spécification)	
Palette	aucune spécification	
Bitmap	Graphique 2D défini dans DAVIC 1.0	1
	MPEG-2 Vidéo Intra-trame (ISO/CEI 13818-2)	2
Text	Le sous-ensemble de l'ISO/CEI 8859-1 comme spécifié dans HTML 2.0	1
EntryField	Le sous-ensemble de l'ISO/CEI 8859-1 comme spécifié dans HTML 2.0	1
HyperText	Le sous-ensemble de l'ISO/CEI 8859-1 comme spécifié dans HTML 2.0	1
Stream	MPEG-2 Système spécifié dans l'ISO/CEI 13818-1	1
	MPEG-1 Audio spécifié dans l'ISO/IEC 11172-3	2
	AIFF-C	3
	flux temps réel spécifié dans DAVIC 1.0	4
LineArt	(aucune spécification)	
CursorShape	(aucune spécification)	
InterchangedProgram	MHEG-6	1

II.5 Registres d'entrée/sortie

Pour en faire une norme d'instanciation qui marche, le domaine applicatif devrait spécifier un ou plusieurs registres *InputEventRegisters*. Chaque registre a un numéro qui est échangé comme un paramètre d'un objet *Scene*. Le contenu d'un registre *InputEventRegister* (qui lui n'est pas échangé) est formé d'un ensemble de numéros (représentant des *UserInputEventTags*) et un nom. Il est recommandé que les noms aient autant de sémantique que possible ("Up" est correct; "13" ne l'est pas). Les paires nom/numéro attachent un *UserInputEventTag* spécifique à un événement d'entrée logique. Il revient au moteur MHEG-5 d'attacher l'événement d'entrée logique à un ou plusieurs événements d'entrée physiques.

Le tableau suivant est un exemple d'ensemble de registres *InputEventRegisters*:

Registre #	***UserInputEventTag***	**Nom**	**Commentaire**
1	1	*Up*	
	2	*Down*	
	3	*Left*	
	4	*Right*	
	5-14	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, respectivement	
	15	*Select*	
	16	*Exit*	
	17	*Help*	
	18-99	réservé pour spécification future.	
	100-	réservé au distributeur	

II.6 Contraintes sémantiques sur les applications MHEG-5

Un domaine applicatif MHEG-5 peut contraindre ses applications à respecter certaines dimensions. Le tableau suivant contient une liste d'attributs de la présente Recommandation qui sont optionnels et peuvent être supportés à des degrés variables. Pour chacun de ces points, une application MHEG-5 doit prendre une décision comme décrit dans la colonne de droite.

Le tableau suivant est un exemple de telles décision:

Attribut	Contrainte
FreeMovingCursor	optionnel
ApplicationStacking	optionnel
Scaling	optionnel
SceneCoordinateSystem(X,Y)	les systèmes de coordonnées suivants sont supportés: 720 × 576, 704 × 576, 640 × 576, 544 × 576, 540 × 576, 480 × 576, 352 × 576, 352 × 288, 720 × 480, 704 × 480, 640 × 480, 544 × 480, 480 × 480, 352 × 480 et 352 × 240.
SceneAspectRatio(W,H)	Les rapports d'aspect suivants sont supportés: 1/1, 4/3 et 16/9
AncillaryConnections	optionnel
TrickModes	optionnel
MultipleRTGraphicsStreams(N)	un objet RTGraphics actif à la fois
MultipleAudioStreams(N)	un flux audio de type *Stream* et un type de mémoire à la fois
MultipleVideoStreams(N)	un flux vidéo déroulé à la fois
OverlappingVisibles(N)	pas de contrainte
Cloning	obligatoire

II.7 (*EngineEvent)* Evénement moteur

Ce domaine applicatif ne réserve aucune valeur particulière d'événement *EngineEvent*. Ceci est laissé à l'initiative du développeur d'application.

II.8 (*GetEngineSupport*) Récupère un soutien du moteur

Les chaînes autorisées pouvant être soumises à l'action *GetEngineSupport* sont celles définies dans la présente Recommandation. En plus de ces chaînes, un domaine applicatif peut en définir d'autres, dans ce cas le domaine applicatif les définira de manière détaillée dans un tableau identique à celui ci-dessus. Cet exemple ne définit aucune chaîne supplémentaire.

II.9 Mappage de protocole et interaction externe

Le tableau suivant fournit des exemples de mappage vers un environnement externe.

Entité MHEG-5	Mappage requis	La sémantique de ces structures MHEG-5 requiert la spécification suivante
OpenConnection, CloseConnection	mappage vers la gestion de connexion	• dans *OpenConnection*: – Protocole: l'une des deux chaînes: "RTPC" ou "RNIS" – adresse: E.164 NSAP
objets *Remote-Program*	mappage vers le protocole d'appel *RemoteProgram* dans le domaine applicatif	• dans *Call* et *Fork*: – *Name* – *Parameters* – *ProgramConnectionTag* sera cohérent avec DSM-CC
espace de nom d'application	mappage vers l'espace de nom du domaine applicatif	• *ObjectReference* • *ContentReference* sera cohérent avec DSM-CC
espace de nom de domaine applicatif lorsqu'une action de transition utilise le paramètre d'étiquette de connexion	mappage vers l'espace de nom du domaine applicatif	• *ObjectReference* • *ContentReference* sera cohérent avec DSM-CC
espace de nom de stockage permanent	mappage vers l'espace de nom de stockage permanent	• dans *StorePersistent* et *ReadPersistent*: – *InFileName, OutFileName* sera cohérent avec DSM-CC
action *Stream*	mappage vers l'espace de flux du domaine applicatif	• dans *Stream* – *Speed* – *CounterPosition*
événement *Stream*	mappage vers les états de flux et les événements de flux dans le domaine applicatif	• dans *Stream* – *StreamPlaying, StreamStopped* (mappage vers la machine d'état de flux du domaine applicatif) – *CounterPosition* – *StreamEventTag*

SERIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
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	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	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
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 pour données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information
Série Z	Langages de programmation