UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T X.753

(10/97)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE X: RÉSEAUX POUR DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

Gestion OSI – Fonctions de gestion et fonctions ODMA

Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: séquenceur de commandes

Recommandation UIT-T X.753

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX POUR DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

RÉSEAUX PUBLICS POUR DONNÉES	
Services et fonctionnalités	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalisation et commutation	X.50–X.89
Aspects réseau	X.90–X.149
Maintenance	X.150–X.179
Dispositions administratives	X.180–X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	
Modèle et notation	X.200–X.209
Définitions des services	X.210–X.219
Spécifications des protocoles en mode connexion	X.220–X.229
Spécifications des protocoles en mode sans connexion	X.230–X.239
Formulaires PICS	X.240–X.259
Identification des protocoles	X.260–X.269
Protocoles de sécurité	X.270–X.279
Objets gérés de couche	X.280–X.289
Tests de conformité	X.290–X.299
INTERFONCTIONNEMENT DES RÉSEAUX	
Généralités	X.300–X.349
Systèmes de transmission de données par satellite	X.350–X.399
SYSTÈMES DE MESSAGERIE	X.400–X.499
ANNUAIRE	X.500–X.599
RÉSEAUTAGE OSI ET ASPECTS SYSTÈMES	
Réseautage	X.600–X.629
Efficacité	X.630–X.639
Qualité de service	X.640–X.649
Dénomination, adressage et enregistrement	X.650–X.679
Notation de syntaxe abstraite numéro un (ASN.1)	X.680–X.699
GESTION OSI	
Cadre général et architecture de la gestion-systèmes	X.700–X.709
Service et protocole de communication de gestion	X.710–X.719
Structure de l'information de gestion	X.720–X.729
Fonctions de gestion et fonctions ODMA	**X.730–X.799**
SÉCURITÉ	X.800–X.849
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850–X.859
Traitement transactionnel	X.860–X.879
Opérations distantes	X.880–X.899
TRAITEMENT RÉPARTI OUVERT	X.900–X.999

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

NORME INTERNATIONALE 10164-21

RECOMMANDATION UIT-T X.753

TECHNOLOGIES DE L'INFORMATION – INTERCONNEXION DES SYSTÈMES OUVERTS – GESTION-SYSTÈMES: SÉQUENCEUR DE COMMANDES

Source

La Recommandation X.753 de l'UIT-T a été approuvée le 24 octobre 1997. Un texte identique est publié comme Norme internationale ISO/CEI 10164-21.

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

NORME INTERNATIONALE

RECOMMANDATION UIT-T

TECHNOLOGIES DE L'INFORMATION – INTERCONNEXION DES SYSTÈMES OUVERTS – GESTION-SYSTÈMES: SÉQUENCEUR DE COMMANDES

1 Domaine d'application

La présente Recommandation | Norme internationale définit une fonction de gestion-systèmes qui peut être utilisée par un processus d'application dans un environnement de gestion centralisé ou décentralisé pour interagir à des fins de gestion-systèmes, selon la définition de la Rec. X.700 du CCITT | ISO/CEI 7498-4. La présente Recommandation | Norme internationale définit le séquenceur de commandes constitué de définitions génériques, de services et d'unités fonctionnelles. Cette fonction est positionnée dans la couche Application de la Rec. UIT-T X.200 | ISO/CEI 7498-1 et définie selon le modèle fourni par l'ISO 9545. Le rôle des fonctions de gestion-systèmes est décrit dans la Rec. UIT-T X.701 | ISO/CEI 10040.

La présente Recommandation | Norme internationale:

- établit les besoins de l'utilisateur pour le séquenceur de commandes;
- établit les modèles qui concernent les services fournis pour les besoins de l'utilisateur;
- définit les services fournis par la fonction;
- spécifie le protocole qui est nécessaire pour fournir les services;
- définit les relations entre ces services et les opérations et notifications SMI;
- définit les relations avec d'autres fonctions de gestion-systèmes;
- spécifie les conditions de conformité;
- définit un langage de description à utiliser dans l'environnement du séquenceur de commandes.

En revanche, la présente Recommandation | Norme internationale ne définit pas:

- la nature de toute réalisation prévue pour assurer la fonction de séquenceur de commandes;
- les modalités d'exécution de la gestion par l'utilisation du séquenceur de commandes;
- la nature de toute instruction qui résulterait de l'utilisation du séquenceur de commandes;
- les services nécessaires à l'établissement d'associations de gestion et à leur libération normale ou anormale.

2 Références normatives

Les Recommandations | Normes internationales suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | Norme internationale. Au moment de la publication, les éditions indiquées étaient en vigueur. Toutes Recommandations et Normes sont sujettes à révision et les parties prenantes aux accords fondés sur la présente Recommandation | Norme internationale sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des Recommandations et Normes indiquées ci-après. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur. Le Bureau de la normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT-T en vigueur.

2.1 Recommandations | Normes internationales identiques

– Recommandation UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: le modèle de référence de base* .

– Recommandation UIT-T X.210 (1993) | ISO/CEI 10731:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: conventions pour la définition des services de l'interconnexion de systèmes ouverts*.

– Recommandation UIT-T X.701 (1992) | ISO/CEI 10040:1992, *Technologies de l'information – Interconnexion des systèmes ouverts – Aperçu général de la gestion des systèmes* .

– Recommandation UIT-T X.710 (1997) | ISO/CEI 9595:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – Service commun de transfert d'informations de gestion* .

– Recommandation UIT-T X.711 (1997) | ISO/CEI 9596-1:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – Spécification du protocole commun de transfert d'informations de gestion*.

– Recommandation X.721 du CCITT (1992) | ISO/CEI 10165-2:1992, *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: Définition des informations de gestion*.

– Recommandation X.722 du CCITT (1992) | ISO/CEI 10165-4:1992, *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: Directives pour la définition des objets gérés*.

– Recommandation UIT-T X.724 (1996) | ISO/CEI 10165-6:1997, *Technologies de l'information – Interconnexion des systèmes ouverts – Structure de l'information de gestion: spécifications et directives pour l'établissement des formulaires de déclaration de conformité d'implémentations associés à la gestion OSI*.

– Recommandation UIT-T X.725 (1995) | ISO/CEI 10165-7:1996, *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: Modèle général de relation* .

– Recommandation X.730 du CCITT (1992) | ISO/CEI 10164-1:1993, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion des systèmes: Fonction de gestion des objets* .

– Recommandation X.731 du CCITT (1992) | ISO/CEI 10164-2:1992, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion des systèmes: Fonction de gestion d'états* .

– Recommandation X.733 du CCITT (1992) | ISO/CEI 10164-4:1992, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion des systèmes: Fonction de signalisation des alarmes* .

– Recommandation X.734 du CCITT (1992) | ISO/CEI 10164-5:1993, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion des systèmes: Fonction de gestion des rapports d'événement*.

– Recommandation UIT-T X.735 (1992) | ISO/CEI 10164-6:1993, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de commande des registres de consignation*.

– Recommandation UIT-T X.739 (1993) | ISO/CEI 10164-11:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion des systèmes: objets et attributs métriques* .

– Recommandation UIT-T X.741 (1995) | ISO/CEI 10164-9:1995, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: objets et attributs de contrôle d'accès* .

– Recommandation UIT-T X.746 (1995) | ISO/CEI 10164-15:1995, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: Fonction de programmation* .

2.2 Paires de Recommandations | Normes internationales équivalentes par leur contenu technique

– Recommandation X.209 du CCITT (1988), *Spécification des règles de codage de base pour la notation de syntaxe abstratie numéro un (ASN.1)*.

ISO/CEI 8825:1990, *Technologies de l'information – Interconnexion des systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro un* (*ASN.1*).

– Recommandation UIT-T X.291 (1995), *Cadre général et méthodologie des tests de conformité d'interconnexion des systèmes ouverts pour les Recommandations sur les protocoles pour les applications de l'UIT-T – Spécification de suite de tests abstraite*.

  ISO/CEI 9646-2:1994, *Technologies de l'information – Interconnexion de systèmes ouverts – Cadre général et méthodologie des tests de conformité OSI*.

– Recommandation UIT-T X.296 (1995), *Cadre général et méthodologie des tests de conformité OSI pour les Recommandations sur les protocoles pour les applications de l'UIT-T – Déclarations de conformité d'instance*.

  ISO/CEI 9646-7:1995, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Essais de conformité – Méthodologie générale et procédures – Partie 7: Déclarations de conformité des mises en œuvre*.

– Recommandation X.700 du CCITT (1992), *Cadre de gestion pour l'interconnexion des systèmes ouverts pour les applications du CCITT*.

  ISO/CEI 7498-4:1989, *Systèmes de traitement de l'information – Interconnexion de systèmes ouverts – Modèle de référence de base – Partie 4: Cadre général de gestion*.

3 Définitions

Pour les besoins de la présente Recommandation | Norme internationale, les définitions suivantes s'appliquent.

3.1 Définitions du modèle de référence de base

La présente Recommandation | Norme internationale utilise les termes suivants définis dans la Rec. UIT-T X.200 | ISO/CEI 7498-1:

a) système ouvert;

b) gestion-systèmes.

3.2 Définition des conventions de service

La présente Recommandation | Norme internationale utilise les termes suivants définis dans la Rec. UIT-T X.210 | ISO/CEI 10731:

– primitive.

3.3 Définitions du cadre général de gestion

La présente Recommandation | Norme internationale utilise les termes suivants définis dans la Rec. X.700 du CCITT | ISO/CEI 7498-4:

a) informations de gestion;

b) objet géré.

3.4 Définitions de la présentation générale de la gestion-systèmes

La présente Recommandation | Norme internationale utilise les termes suivants définis dans la Rec. X.701 du CCITT | ISO/CEI 10040:

a) rôle d'agent;

b) objet support de gestion;

c) classe d'objets gérés;

d) rôle de gestionnaire;

e) notification;

f) unité fonctionnelle de gestion-systèmes;

g) opération de gestion-systèmes.

3.5 Définitions du service commun d'information de gestion

La présente Recommandation | Norme internationale utilise les termes suivants définis dans la Rec. UIT-T X.710 | ISO/CEI 9595:

a) attribut;

b) services communs d'information de gestion.

3.6 Définitions supplémentaires

Les termes suivants sont définis dans la présente Recommandation | Norme internationale:

3.6.1 séquenceur de commandes: objet support de gestion représentant une ressource qui fonctionne dans un rôle de gestionnaire comme une destination de notification et comme un initiateur d'opérations déterminé par ses scripts de lancement, jouissant de la capacité de déléguer des activités de gestion.

3.6.2 script de lancement: objet géré qui représente les instructions à exécuter par un séquenceur de commandes.

3.6.3 tâche élémentaire: objet géré qui représente l'exécution d'un script de lancement. Les résultats de l'exécution ou les erreurs d'exécution du script de lancement sont retournés par la tâche élémentaire.

3.6.4 tâche élémentaire interruptible: est dérivée de la classe d'objets gérés tâche élémentaire. Ces tâches élémentaires sont générées de manière dynamique par des pas de lancement asynchrones. Elles peuvent être interrompues par une action "suspendre" qui leur est envoyée et remises en service par une action "reprendre" qui leur est envoyée.

3.6.5 pas de lancement: objet support de gestion auquel un déclencheur peut être envoyé pour entamer l'exécution d'un script de lancement. Un pas de lancement est utilisé comme objet géré de valeur initiale pour une tâche élémentaire.

3.6.6 pas de lancement asynchrone: est dérivé de l'objet pas de lancement. Il renvoie une notification de résultat de déclenchement sans attendre les résultats d'exécution des scripts de lancement. Les résultats d'exécution ou les erreurs d'exécution du script de lancement sont directement notifiés à partir de la tâche élémentaire.

3.6.7 pas de lancement synchrone: est dérivé de l'objet pas de lancement. Il renvoie une notification de résultat de déclenchement ou une alarme sur erreur de traitement après avoir obtenu tous les résultats d'exécution et les erreurs des tâches élémentaires, une fois que ces dernières ont achevé leur exécution.

3.6.8 activateur de déclenchement: initiateur d'exécution de script en faisant en sorte qu'un pas de lancement génère de manière dynamique une ou plusieurs tâches élémentaires. Il dirige une commande vers un pas de lancement sous la forme d'un programmateur d'opérations, de notifications ou d'une action locale.

3.6.9 commande: instruction relative à une activité de gestion qui est effectuée dans le système interne de l'agent conformément au contenu du script de lancement. Une commande est décrite au moyen d'un langage de script. Actuellement, le langage de script de la gestion-systèmes est défini dans l'Annexe F.

4 Abréviations

Pour les besoins de la présente Recommandation | Norme internationale, les abréviations suivantes sont utilisées:

ASN.1 Notation de syntaxe abstraite numéro un (*abstract syntax notation one*)

CMIS Service commun d'information de gestion (*common management information service*)

CS Séquenceur de commandes (*command sequencer*)

IVMO Objet géré de valeur initiale (*initial value managed object*)

OSI Interconnexion des systèmes ouverts (*open systems interconnection*)

LP Pas de lancement (*launch pad*)

SMSL Langage de description de gestion-systèmes (*systems management scripting language*)

5 Conventions

La présente Spécification définit les services destinés au séquenceur de commandes selon les conventions descriptives définies dans la Rec. UIT-T X.210 | ISO/CEI 10731. L'article 9 associe à la définition de chaque service un tableau qui recense les paramètres de service. Pour une primitive de service donnée, la présence de chaque paramètre est décrite par l'une des valeurs suivantes:

M Le paramètre est obligatoire.

(=) La valeur du paramètre est égale à celle du paramètre de la colonne de gauche.

U L'utilisation de ce paramètre est une option proposée à l'utilisateur du service.

– Paramètre non présent dans l'interaction décrite par la primitive en question.

C Paramètre conditionnel. Les conditions sont définies dans le descriptif du paramètre.

P Paramètre soumis aux contraintes imposées par la Rec. UIT-T X.710 | ISO/CEI 9595.

NOTE – Les paramètres marqués "P" dans les tableaux de service de la présente Spécification sont mappés directement avec les paramètres correspondants de la primitive de service CMIS sans modifier la sémantique ou la syntaxe des paramètres.

La police de caractère utilisée pour GDMO (directives pour la définition des objets gérés), ASN.1 (notation de syntaxe abstraite numéro 1) et GRM (modèle général de relation) dans la présente Recommandation | Norme internationale est la police de notation Courier. Le BNF pour le langage SMSL (langage de description de gestion-systèmes) décrit en F.14.11 est en Courier Nouveau. Dans les Annexes F et G, les paramètres de fonction SMSL sont en italique.

6 Prescriptions

Les prescriptions à satisfaire sont les suivantes:

- Besoins de l'utilisateur:
  - permettre la délégation des activités de gestion;
  - réduire la quantité d'échanges nécessaires entre gestionnaire et agents;
  - permettre le fonctionnement des systèmes délégués par le gestionnaire sur les systèmes de l'agent lorsque les échanges entre un gestionnaire et les systèmes de l'agent ont été interrompus ou sont impossibles;
  - fournir un contrôle souple des activités de gestion;
  - fournir un langage de description des procédures nécessaires à l'exécution des opérations de gestion;
  - permettre aux systèmes délégués d'exécuter des opérations CMIS de manière séquentielle.
- Prescriptions opérationnelles:
  - permettre l'exécution préprogrammée ou différée d'une opération de gestion-systèmes;
  - disposer des capacités nécessaires à la modification de la demande d'exécution préprogrammée ou différée;
  - disposer des capacités nécessaires pour lancer, suspendre, reprendre et terminer l'exécution d'opérations de gestion-systèmes sur la base d'actions de gestion temporelle ou sur la base d'apparition d'événements;
  - disposer des capacités nécessaires pour signaler et enregistrer le résultat d'une exécution préprogrammée ou différée;
  - permettre d'envoyer des notifications en cas de changements d'état.

7 Modèle

7.1 Description du modèle

Le modèle décrit la manière dont l'exécution déclenchée, préprogrammée ou différée d'opérations de gestion-systèmes peut être exécutée par le séquenceur de commandes. II décrit également les composantes conceptuelles, la relation entre ces composantes ainsi qu'une description des états et transitions d'état possibles.

La Figure 1 indique une description schématique de la capacité du séquenceur de commandes d'un système donné.

La fonctionnalité d'un séquenceur de commandes est modélisée par les objets suivants: script de lancement, tâche élémentaire et pas de lancement. Il s'agit d'une abstraction OSI de l'exécution d'une opération préprogrammée ou différée dans des systèmes ouverts. Il est admis qu'un séquenceur de commandes contienne n'importe quel nombre de pas de lancement pour lesquels il sert de fournisseur du service. Chaque pas de lancement peut exécuter un script de lancement ou plusieurs scripts de lancement à la fois. Lorsqu'il reçoit un déclenchement en provenance d'un activateur de déclenchement, un pas de lancement initie l'exécution d'un script de lancement. Outre la composante id du déclencheur, le déclencheur peut spécifier un nom de script de lancement (identificateur du script) et insérer, dans le script, des arguments en tant que paramètres dans la composante Liste de paramètres d'exécution.

Il existe deux types de pas de lancement: le pas de lancement asynchrone et le pas de lancement synchrone. Un pas de lancement asynchrone renvoie une notification de résultat de déclenchement sans attendre les résultats d'exécution des scripts de lancement. Les résultats d'exécution ou les erreurs dues à des exécutions du script de lancement sont directement notifiés à partir de la tâche élémentaire. Par contre, un pas de lancement synchrone renvoie une notification de résultat de déclenchement ou une alarme sur erreur de traitement, après avoir obtenu tous les résultats d'exécution et erreurs transmis par les tâches élémentaires, une fois que ces dernières ont achevé leur exécution.

Une instance de script de lancement peut contenir n'importe quel nombre d'instructions unitaires. La composante liste de paramètres d'exécution du déclencheur est une liste de scripts (identifiés par leur identificateur de script) à accomplir ainsi que les paramètres d'entrée correspondants, nécessaires à l'exécution de ces scripts. Une liste de paramètres d'exécution par défaut peut être spécifiée pour qu'un pas de lancement s'exécute en cas de réception d'un déclencheur dans lequel les paramètres de déclenchement ne sont pas spécifiés. Si le pas de lancement n'est pas configuré pour exécuter une liste de paramètres d'exécution par défaut et que la composante liste de paramètres d'exécution ne soit pas fournie par le déclencheur et si le pas de lancement reçoit un déclencheur qui tente de l'activer, un code d'erreur "pas de script" est renvoyé dans le champ "code d'erreur" de la notification de résultat de déclenchement. Le pas de lancement dispose d'un attribut liste des scripts qui peut être configuré pour identifier les scripts qu'il peut exécuter. Si une composante liste de paramètres d'exécution est présente dans le déclencheur, le pas de lancement vérifie si chaque identificateur de script de la composante paramètre d'exécution est présent dans son attribut liste des scripts disponibles. Seules les instances des scripts indiquées par l'identificateur de script et présentes dans l'attribut liste des scripts disponibles sont exécutées. Si aucun identificateur de script n'est présent dans la liste des scripts disponibles, le pas de lancement renvoie un code d'erreur de script rejeté dans le résultat de déclenchement et l'exécution du script n'a pas lieu.

Des classes d'objets langage de description dédié sont dérivées de la classe d'objets script de lancement. Par conséquent, il est admis de spécifier ces instructions comme des instances de script dédiées. Plusieurs ensembles d'instructions de script de lancement peuvent être exécutés séquentiellement ou en parallèle par des tâches élémentaires conformément avec le type de données du paramètre d'exécution. Plusieurs niveaux imbriqués de sous-tâche élémentaire peuvent être nécessaires pour l'exécution d'instances de script.

Pour initier le comportement d'exécution d'instances de script de lancement, un déclencheur peut être envoyé à l'instance d'objet pas de lancement. Des déclencheurs non paramétrés peuvent activer le pas de lancement lorsque le pas de lancement dispose d'une liste de paramètres d'exécution par défaut.

Le pas de lancement agit comme un IVMO pour une tâche élémentaire et fournit la liste de paramètres d'exécution à l'attribut d'exécution des paramètres de la tâche élémentaire. La liste de paramètres d'exécution peut être un seul paramètre d'exécution, une séquence de paramètres d'exécution ou un ensemble de paramètres d'exécution. Le paramètre d'exécution est une séquence d'ids de script et de paramètres de script. L'id de script identifie le nom de l'instance objet géré de l'instance objet de description à exécuter et les paramètres de scripts fournissent les valeurs de paramètre qui doivent être entrées dans l'instance objet de description. Si une séquence de paramètres d'exécution est spécifiée, le pas de lancement génère une tâche élémentaire de manière dynamique afin d'exécuter les instances de script et fournit l'id de script ainsi que ses paramètres (si nécessaire) à partir des paramètres d'exécution de la tâche élémentaire. Une fois la tâche élémentaire achevée, cette opération recommence pour les autres ids de script qui se présentent en séquence dans la liste. Si un ensemble de paramètres d'exécution est spécifié, le pas de lancement fournit l'ensemble d'ids de script ainsi que les paramètres correspondants (si nécessaire) aux tâches élémentaires et les instances de script concernées sont exécutées en parallèle. La sémantique des paramètres échangés entre le pas de lancement et les tâches élémentaires dépend du mécanisme de transmission des paramètres pris en charge par le langage de description dans lequel le script est écrit.

L'exécution d'une instance de script est attribuée à une tâche élémentaire. Cette tâche élémentaire peut générer de manière dynamique d'autres tâches élémentaires si nécessaire. Ceci peut avoir lieu lorsqu'une instance de script invoque d'autres instances de script. Dans ce cas, la tâche élémentaire qui exécute un script d'appel génère de manière dynamique une sous-tâche élémentaire et transmet à la sous-tâche élémentaire l'id de script et les paramètres de script (si nécessaire) du script appelé. La sémantique d'échange de paramètres entre les tâches élémentaires et les sous-tâches élémentaires dépend du mécanisme de transmission des paramètres pris en charge par le langage de description dans lequel le script est écrit.

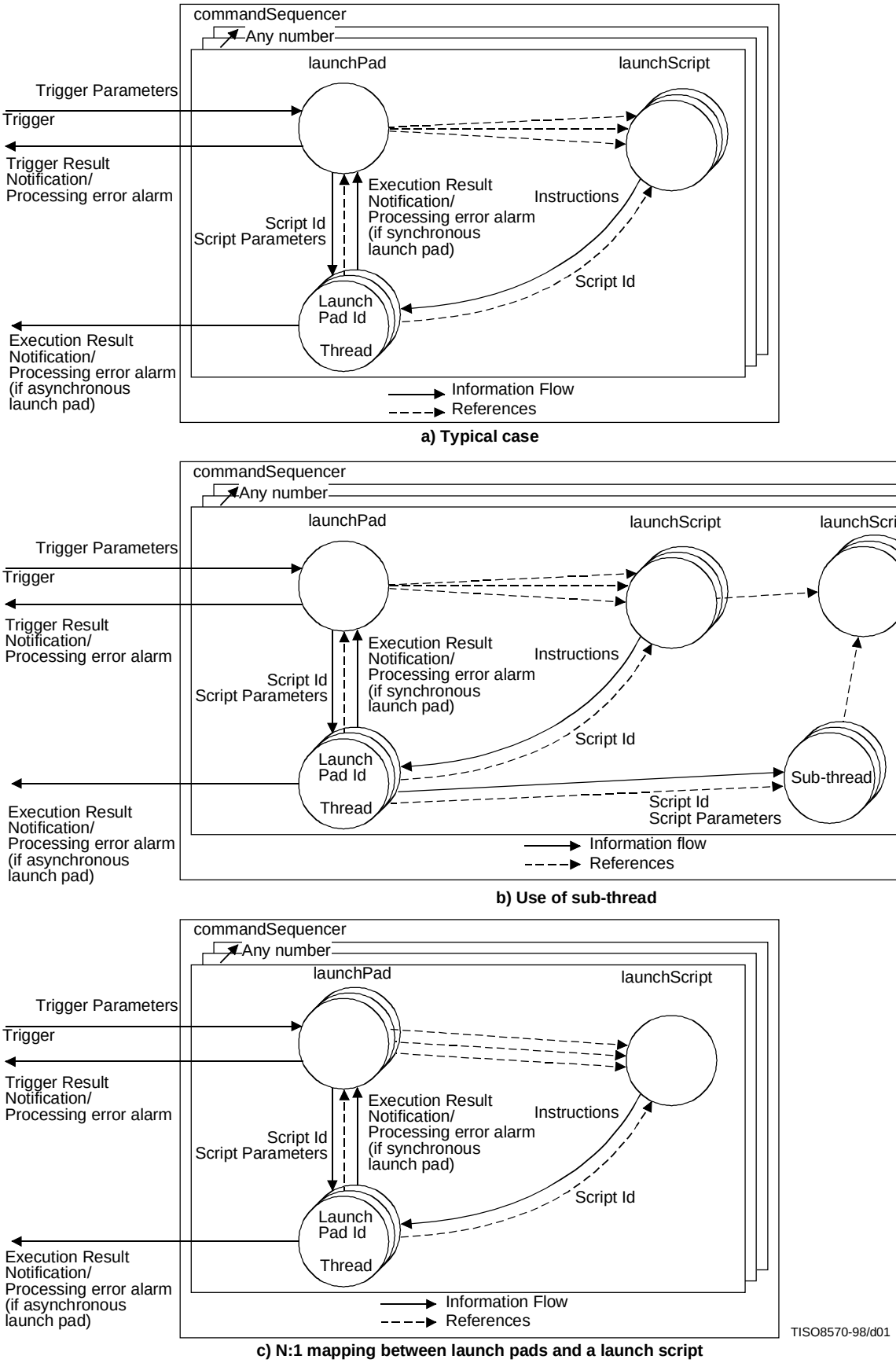


Figure 1 – Modèle de séquenceur de commandes

Il convient que des pas de lancement asynchrones génèrent de manière dynamique des tâches élémentaires interruptibles. Une tâche élémentaire interruptible peut être suspendue et remise en service par des actions "suspendre" et "reprendre" respectivement. Il est admis que des tâches élémentaires particulières, générées par des pas de lancement synchrones ne soient pas suspendues et remises en service. Toutes les tâches élémentaires liées à l'exécution d'un script peuvent être, dans les deux cas, suspendues ou remises en service.

Une tâche élémentaire est achevée une fois que toutes ses sous-tâches élémentaires ont terminé leur exécution avec succès ou ont indiqué une erreur. Dans ce cas, l'exécution d'un script de lancement est terminée; le pas de lancement correspondant renvoie alors un état inactif (au repos). Une tâche élémentaire est contenue par l'objet qui l'a générée de manière dynamique. Une tâche élémentaire peut être contenue par un pas de lancement ou par une autre tâche élémentaire.

Des pas de lancement multiples peuvent faire référence à un script de lancement particulier. Des tâches élémentaires multiples peuvent faire référence au même script de lancement. L'existence d'un script donné est indépendante des éventuels pas de lancement ou tâches élémentaires qui y font référence. Des scripts de lancement sont définis dans les classes de scripts dédiées dérivées de la classe d'objets script de lancement. La sémantique et la syntaxe de ces scripts sont précisées dans la définition du langage de description dans lequel les scripts sont écrits. La définition du langage de description spécifie également un ensemble de fonctions fondamentales de description qui sont nécessaires pour que les scripts de lancement disposent de la capacité de contrôle et de traitement requise.

Il convient d'utiliser la classe d'objets gérés script de chaîne générale pour écrire des scripts représentés sous la forme d'une chaîne générale. L'attribut nom de langage d'écriture de script indique le nom de ce langage de description et l'attribut contenu du script représente le script écrit dans le langage de description sous la forme d'une chaîne générale. Les Annexes F et G définissent un langage de description dédié, le langage de description de gestion-systèmes (SMSL, *system management scripting language*) comme étant le langage de description qu'il conviendrait d'utiliser pour écrire des scripts représentés sous la forme d'une chaîne générale. Il est possible de définir d'autres classes de scripts. L'Annexe D définit un langage d'écriture de script de transfert d'information de gestion (cmisScript), un langage de description qui se présente sous la forme d'objets gérés qui peuvent être utilisés pour écrire des scripts dans l'environnement CMIS.

7.2 Processus de déclenchement et compte rendu de résultats

Les activateurs de déclenchement envoyés au pas de lancement peuvent prendre diverses formes telles que des programmateurs, des opérations, des notifications et des actions locales. Lorsqu'un pas de lancement reçoit un déclencheur, il génère de manière dynamique une ou plusieurs tâches élémentaires afin d'exécuter un script. Une fois générées de manière dynamique toutes les tâches élémentaires liées à un déclencheur donné, un pas de lancement asynchrone émet une notification de résultat de déclenchement qui comprend des ensembles d'ids de tâches élémentaires et d'ids de script. Les résultats d'exécution de script sont directement envoyés au gestionnaire par les tâches élémentaires en tant que notifications de résultats d'exécution, dans le cas du pas de lancement asynchrone. Une fois terminées toutes les tâches élémentaires liées à un déclencheur donné, un pas de lancement synchrone synchronise tous les résultats d'exécution ou les erreurs en provenance des tâches élémentaires et envoie au gestionnaire une notification de résultat de déclenchement qui comprend des ensembles d'ids de tâches élémentaires, d'ids de script et de résultats d'exécution ou d'erreurs.

L'attribut type de résultat d'exécution du script identifie le type de résultat attendu de l'exécution du script et il convient qu'il corresponde à l'attribut type de résultat d'exécution du résultat d'exécution. Le champ errorCode (code d'erreur) du résultat d'exécution est mis sur le code "pas d'erreur" lorsque l'exécution s'est achevée avec succès; autrement, il est mis sur le code d'erreur approprié.

Une tâche élémentaire en cours d'exécution se termine spontanément soit après achèvement de son exécution, soit dans des conditions anormales. Dans ce dernier cas, cette tâche élémentaire indique une terminaison anormale en émettant une notification d'alarme d'erreur de traitement.

Les notifications de résultat d'exécution et d'alarme d'erreur de traitement sont émises par la tâche élémentaire et transmises à la ou aux destinations de notification appropriées. Dans le cas d'un pas de lancement asynchrone, ces notifications sont transmises vers les destinations de notification externes tandis que, dans le cas d'un pas de lancement synchrone, ces notifications sont envoyées au pas de lancement.

Un gestionnaire peut volontairement arrêter tous les processus de lancement au moyen d'une opération delete (supprimer) transmise au pas de lancement correspondant. Sur réception d'une opération delete (suppression), si une association de noms tâche élémentaire-pas de lancement comprend une définition "DELETE DELETES-CONTAINED-OBJECTS", toutes ses tâches élémentaires qui donnent lieu à l'exécution du script sont arrêtées et supprimées. Le pas de lancement est ensuite supprimé.

Un gestionnaire peut volontairement arrêter toutes les exécutions liées à une tâche élémentaire par une opération delete (suppression) transmise à la tâche élémentaire correspondante. Une fois reçue une opération delete, si l'association de noms tâche élémentaire-tâche élémentaire comprend une définition "DELETE DELETES-CONTAINED-OBJECTS" (suppression supprime objet contenu), toutes ses sous-tâches élémentaires liées à l'exécution du script sont arrêtées et supprimées. La tâche élémentaire est ensuite supprimée.

Pour obtenir l'arrêt de l'exécution de tous les scripts en cours d'exécution par un pas de lancement synchrone ou asynchrone, une action "terminer" peut être envoyée au pas de lancement. Toutes les tâches élémentaires liées à l'exécution des scripts sont arrêtées et supprimées lorsqu'une action "terminer" est reçue par le pas de lancement.

Le lancement de toutes les tâches élémentaires en cours d'exécution par un pas de lancement synchrone ou asynchrone peut être suspendu par une action "suspendre" envoyée au pas de lancement et peut ensuite être remis en service par une action "reprendre".

L'exécution de scripts par des tâches élémentaires interruptibles générées de manière dynamique par un pas de lancement peut être suspendue par une action "suspendre" envoyée à la tâche élémentaire puis ensuite remise en service par une action "reprendre". Il convient que l'id de tâche élémentaire, renvoyé dans la notification de résultat de déclenchement, soit fourni en tant que paramètre pour suspendre et reprendre des actions.

Le pas de lancement dispose d'attributs permettant de surveiller un attribut spécifique d'une instance d'objets spécifiques. Si la valeur de l'attribut surveillé change, un déclencheur est généré pour obtenir l'exécution d'une liste de scripts spécifiés.

Si l'attribut surveillé est un compteur de discrimination d'événement (EDC, *event discrimination counter*) tel que défini dans l'Annexe C, les notifications filtrées par l'EDC déclenchent le lancement des scripts par le pas de lancement.

7.3 Gestion de séquenceur de commandes

Les valeurs d'attribut du pas de lancement, de la tâche élémentaire, du script de lancement et des instances d'objets gérés de description dédiée sont récupérées et modifiées par le biais des opérations Get (extraction) et Set (affectation) respectivement.

Les Tableaux 1 à 5 suivants mappent entre les attributs d'état du séquenceur de commandes, du pas de lancement, de la tâche élémentaire et des objets gérés script et les états définis dans la Rec. X.731 du CCITT | ISO/CEI 10164-2.

NOTE – "–" signifie n'importe quelle valeur.

Tableau 1 – Tableau d'état du séquenceur de commandes

Etat du séquenceur de commandes	Etat administratif	Etat opérationnel
CS non opérationnel	–	désactivé
CS opérationnel	déverrouillé	activé
CS verrouillé	verrouillé	activé
CS en cours d'arrêt	en cours d'arrêt	activé

Lorsqu'un séquenceur de commandes est à l'état opérationnel désactivé, il est dans un état totalement inexploitable et ses pas de lancement n'exécutent pas de scripts. S'il est à l'état activé, un événement qui consiste à exécuter une opération à la limite de l'objet géré peut entraîner sa transition de l'état administratif verrouillé à l'état déverrouillé et vice versa. Lorsque le séquenceur de commandes passe à l'état verrouillé, ses pas de lancement interrompent l'exécution des scripts de lancement. Par contre, lorsqu'il passe à l'état administratif déverrouillé, les pas de lancement sont prêts à démarrer ou à reprendre l'exécution des scripts de lancement.

Lorsqu'un pas de lancement est à l'état opérationnel désactivé, il est dans un état totalement inexploitable et ne peut exécuter de scripts. S'il est à l'état activé, un événement qui consiste à exécuter une opération à la limite de l'objet géré peut entraîner sa transition de l'état administratif verrouillé à l'état déverrouillé et vice versa. Lorsque le pas de lancement passe à l'état verrouillé, il suspend l'exécution de scripts de lancement. Par contre, lorsqu'il passe à l'état administratif déverrouillé, les pas de lancement sont prêts à démarrer ou à reprendre l'exécution des scripts de lancement. Le pas de lancement est désactivé par un processus de commande interne, suivant un calendrier prédéterminé et sa valeur d'état de disponibilité est hors service. Une action "suspend" (suspendre) entraîne le passage de l'état de commande à "suspendu" et une action "reprendre" le refait passer à son état par défaut, c'est-à-dire vide.

Tableau 2 – Tableau d'état du pas de lancement

Etat du pas de lancement	Etat administratif	Etat opérationnel	Etat de commande	Etat d'utilisation	Etat de disponibilité
LP non opérationnel	–	désactivé	–	–	
LP opérationnel	déverrouillé	activé	–	occupé	
LP opérationnel	déverrouillé	activé	–	au repos	–
LP verrouillé	verrouillé	activé	–	au repos	–
LP en service	–	–	–	–	non hors service
LP hors service	–	–	–	–	hors service
LP interrompu	–	–	interrompu	–	–
LP remis en service	–	–	vide	–	–

Tableau 3 – Tableau d'état de tâche élémentaire

Etat de la tâche élémentaire	Etat opérationnel
Tâche élémentaire non opérationnelle	désactivé
Tâche élémentaire opérationnelle	activé

La tâche élémentaire est à l'état activé lorsqu'elle effectue une exécution de script et à l'état désactivé dans le cas contraire.

Tableau 4 – Tableau d'état de tâche élémentaire interruptible

Etat de la tâche élémentaire interruptible	Etat opérationnel	Etat de commande
Tâche élémentaire interruptible non opérationnelle	désactivé	–
Tâche élémentaire interruptible opérationnelle	activé	–
Tâche élémentaire interruptible suspendue	–	suspendu
Tâche élémentaire interruptible remise en service	–	vide

Une action "suspend" (suspendre) fait passer l'état de commande de la tâche élémentaire interruptible à "suspendu" et une action "reprendre" le refait passer à sa valeur par défaut, c'est-à-dire vide.

Tableau 5 – Tableau d'état du script de lancement

Etat du script de lancement	Etat administratif
Exécution LS autorisée	déverrouillé
Exécution LS non autorisée	verrouillé

Un événement qui consiste à exécuter une opération à la limite de l'objet géré peut entraîner une transition du script de l'état administratif verrouillé à l'état déverrouillé et vice versa. Lorsque le script de lancement passe à l'état verrouillé, il ne peut pas être exécuté par un pas de lancement autre que ceux qui sont en train de l'exécuter. Par contre, lorsqu'il passe à l'état administratif déverrouillé, le script de lancement est prêt pour exécution par d'autres pas de lancement.

7.4 Programmation du séquenceur de commandes

Un bloc de programmation du programmateur externe permet de programmer l'activation, par les déclencheurs, des pas de lancement du séquenceur de commandes. L'attribut état de disponibilité du pas de lancement passera à l'état "hors service" ou "non hors service" selon les caractéristiques de programmation spécifiées par un objet géré programmateur externe. La sémantique du bloc de programmation du programmateur externe est décrite dans la Rec. X.734 du CCITT | ISO/CEI 10164-5 ainsi que dans la Rec. X.735 du CCITT | ISO/CEI 10164-6.

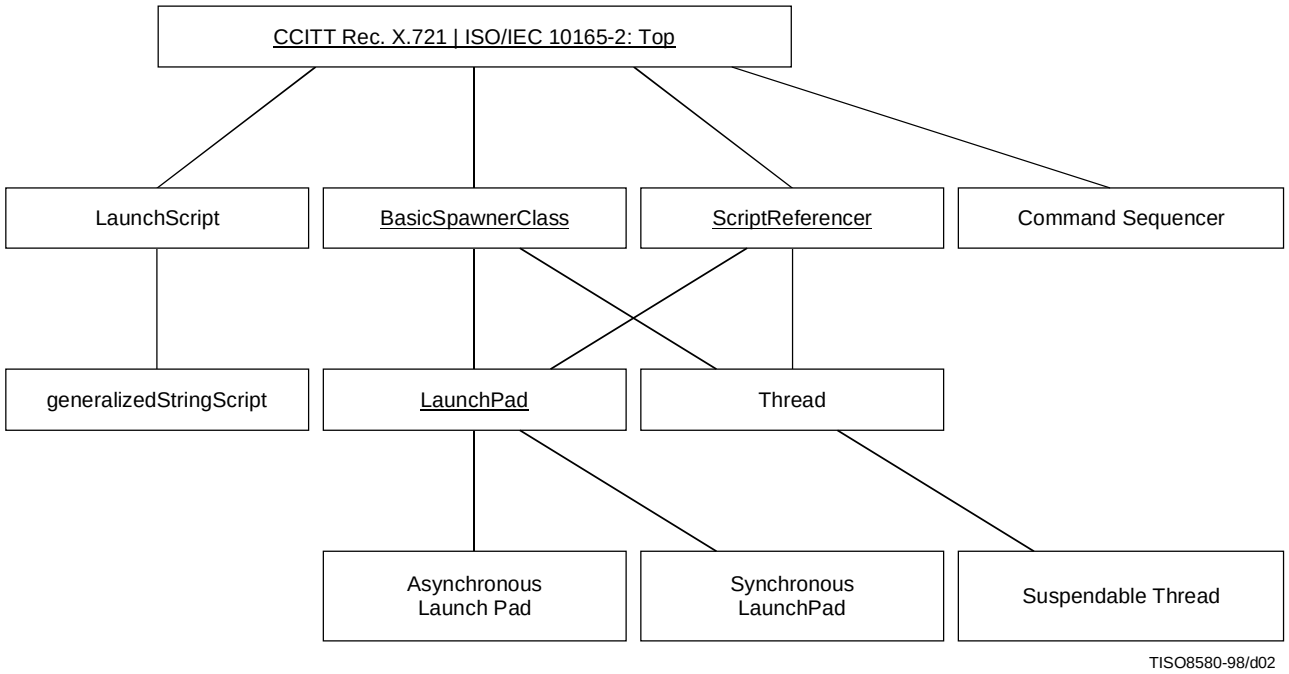
7.5 Commande d'accès

Il convient que le séquenceur de commandes autorise l'accès aux instances d'objets gérés qui sont exploitées par ses tâches élémentaires. Il convient que le comportement de la tâche élémentaire, lorsqu'une opération sur une instance donnée est refusée, soit défini dans le script. Par exemple, si une opération est refusée, une erreur administrative d'accès non autorisé peut être renvoyée par une notification d'alarme d'erreur de traitement.

8 Définitions génériques

8.1 Objets gérés

La présente Spécification définit un ensemble de classes d'objets gérés. La Figure 2 montre la structure d'héritage de ces classes d'objets gérés.



NOTE – Les classes d'objets susceptibles de ne pas être instanciées sont soulignées.

Figure 2 – Structure d'héritage des ressources d'un séquenceur de commandes

La structure de confinement de ces classes d'objets gérés est illustrée à la Figure 3.

8.1.1 Séquenceur de commandes

8.1.1.1 Aperçu général

- Objet support de gestion représentant une ressource qui agit en rôle de gestionnaire en tant qu'invocateur d'opérations déterminées par ses scripts de lancement et en tant que destination de notification.
- Agit en tant que fournisseur du service pour un pas de lancement.
- Contient un ou plusieurs pas de lancement.

8.1.1.2 Blocs de l'objet support de gestion de séquenceur de commandes

L'objet support de gestion de séquenceur de commandes contient le bloc obligatoire suivant:

- bloc séquenceur de commandes.

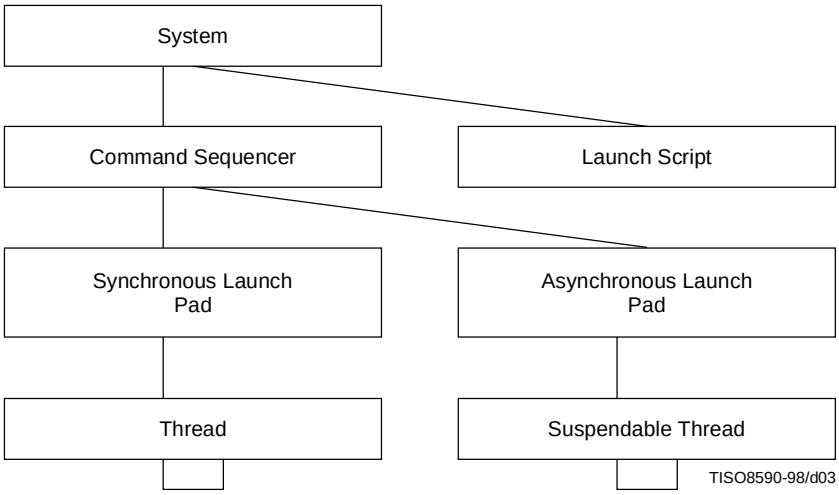


Figure 3 – Structure de confinement des ressources de séquenceur de commandes

8.1.1.3 Caractéristiques du séquenceur de commandes

La classe d'objets support de gestion du séquenceur de commandes a les attributs suivants.

8.1.1.3.1 Id de séquenceur de commandes

La valeur de cet attribut définit une instance de la classe d'objets support de gestion du séquenceur de commandes.

8.1.1.3.2 Etat administratif

Cet attribut représente la capacité administrative du séquenceur de commandes à exécuter sa fonction. Les états administratifs suivants sont définis:

a) déverrouillé – Les pas de lancement du séquenceur de commandes sont autorisés à démarrer ou à reprendre l'exécution des scripts de lancement;

b) verrouillé – Les pas de lancement du séquenceur de commandes ne sont pas autorisés à démarrer l'exécution des scripts de lancement. Si des exécutions sont en cours, elles sont suspendues;

c) en cours d'arrêt – Le séquenceur de commandes est en cours d'arrêt et ses pas de lancement ne seront pas autorisés à effectuer d'éventuelles exécutions de script.

8.1.1.3.3 Etat opérationnel

Cet attribut représente la capacité opérationnelle du séquenceur de commandes à remplir ses fonctions.

Les états opérationnels suivants sont définis:

a) activé – Le séquenceur de commandes est opérationnel et prêt à être utilisé;

b) désactivé – Le séquenceur de commandes n'est pas disponible pour utilisation.

8.1.1.4 Notifications du séquenceur de commandes

La classe d'objets gérés support de séquenceur de commandes dispose des notifications suivantes:

– création d'objets comme défini dans la Rec. X.730 du CCITT | ISO/CEI 10164-1;

– suppression d'objets comme défini dans la Rec. X.730 du CCITT | ISO/CEI 10164-1;

– changement d'état comme défini dans la Rec. X.730 du CCITT | ISO/CEI 10164-1.

8.1.2 Tâche élémentaire

8.1.2.1 Aperçu général

– Objet géré qui modélise l'exécution des commandes et les exécute conformément à un script.

– Sous-classe des classes d'objets gérés, générateur de base et référenceur de script.

– Chaque tâche élémentaire est dédiée à un pas de lancement synchrone.

– Cette classe d'objets a une notification qui donne le résultat d'exécution du script de lancement.

8.1.2.2 Blocs de la classe d'objets gérés tâche élémentaire

La classe d'objets gérés tâche élémentaire contient les blocs obligatoires suivants:

- bloc tâche élémentaire;
- bloc résultat d'exécution.

8.1.2.3 Caractéristiques de la classe d'objets gérés tâche élémentaire

La classe tâche élémentaire a les attributs suivants.

8.1.2.3.1 Id de tâche élémentaire

La valeur de cet attribut identifie une instance de la classe d'objets gérés tâche élémentaire.

8.1.2.3.2 Id de script

La valeur de cet attribut identifie une instance de classe d'objets gérés script de lancement qui est en cours d'exécution.

8.1.2.3.3 Paramètres d'exécution

La composante paramètres de script de cet attribut est une liste de valeurs de paramètres fournie par un pas de lancement et qui est nécessaire à l'exécution d'un script. La composante Id de script de cet attribut indique le ou les scripts qu'il convient d'exécuter avec les paramètres correspondants.

8.1.2.3.4 Etat opérationnel

Cet attribut représente la capacité opérationnelle de la tâche élémentaire à remplir ses fonctions. Les états opérationnels suivants sont définis:

a) activé – La tâche élémentaire est opérationnelle et est en train d'effectuer l'exécution d'un script;

b) désactivé – La tâche élémentaire n'est pas opérationnelle.

8.1.2.4 Notifications de la classe d'objets gérés tâche élémentaire

La classe d'objets gérés tâche élémentaire dispose des notifications suivantes qu'elle envoie à la classe d'objets gérés pas de lancement:

- résultat d'exécution comme défini au 8.2.1;
- alarme d'erreur de traitement comme défini dans la Rec. X.733 du CCITT | ISO/CEI 10164-4.

8.1.3 Tâche élémentaire interruptible

8.1.3.1 Aperçu général

- Sous-classe de la classe d'objets gérés tâche élémentaire.
- Générée de manière dynamique par un pas de lancement asynchrone.
- Peut être suspendue et remise en service au moyen d'actions suspendre et reprendre.

8.1.3.2 Blocs de la tâche élémentaire interruptible

Les blocs obligatoires de la classe d'objets gérés tâche élémentaire interruptible sont les suivants:

- accepteur d'action suspendre/reprendre.

8.1.3.3 Caractéristiques de la tâche élémentaire interruptible

La tâche élémentaire interruptible dispose des attributs suivants.

8.1.3.3.1 Etat de commande

L'attribut état de commande est défini dans la Rec. X.731 du CCITT | ISO/CEI 10164-2. La valeur par défaut est vide. Si l'exécution est suspendue, l'état de commande devient "suspendu" et si l'exécution est reprise, la valeur devient vide.

8.1.3.4 Actions de la classe d'objets gérés tâche élémentaire interruptible

Les actions suivantes peuvent être envoyées à la classe d'objets gérés tâche élémentaire interruptible:

- suspendre;
- reprendre.

8.1.4 Script de lancement

8.1.4.1 Aperçu général

- Information de gestion qui représente une série d'instructions dans des langages de description dédiés.
- Il est recommandé que chaque script soit indépendant.

8.1.4.2 Caractéristiques du script de lancement

La classe script de lancement dispose des attributs suivants.

8.1.4.2.1 Id de script

La valeur de cet attribut identifie une instance de la classe d'objets gérés script de lancement.

8.1.4.2.2 Type de résultat d'exécution

La valeur de cet attribut identifie le type de résultat d'exécution attendu.

8.1.4.2.3 Etat administratif

Cet attribut est défini dans la Rec. X.731 du CCITT | ISO/CEI 10164-2.

8.1.4.3 Blocs de la classe d'objets script de lancement

La classe d'objets gérés script de lancement contient le bloc obligatoire suivant:

- bloc script de lancement.

8.1.5 Classe générateur de base

8.1.5.1 Aperçu général

- Signifie la capacité de création de nouvelles instances d'objets contenus avec génération automatique de noms pour ces nouvelles instances.
- Hyperclasse de classes d'objets pas de lancement et tâche élémentaire de séquence de commandes.

8.1.5.2 Blocs de la classe de générateur de base

La classe générateur de base contient les blocs obligatoires suivants:

- bloc générateur de base.

8.1.6 Pas de lancement

8.1.6.1 Aperçu général

- Sous-classe des classes d'objets gérés générateur de base et référenceur de script.
- Initiateur de l'exécution du script de lancement après réception d'un déclencheur.
- Agit comme un objet IVMO pour une tâche élémentaire donnée.
- Un script est exécuté par le pas de lancement au moyen d'une ou de plusieurs tâches élémentaires.

8.1.6.2 Blocs de la classe d'objets gérés pas de lancement

Les blocs obligatoires de la classe d'objets gérés pas de lancement sont les suivants:

- bloc pas de lancement;
- accepteur action de déclenchement;
- passeur de paramètres;
- bloc résultat de déclenchement;
- accepteur événement de déclenchement;
- accepteur opération terminée;
- programmateur externe;
- accepteur action suspendre/reprendre.

8.1.6.3 Caractéristiques du pas de lancement

La classe d'objets gérés pas de lancement dispose des attributs suivants.

8.1.6.3.1 Liste de scripts disponibles

Il s'agit d'une liste de tous les scripts qu'un pas de lancement est capable d'exécuter. Le pas de lancement exécute uniquement les scripts qui sont présents à la fois dans la composante liste de paramètres d'exécution des paramètres de déclenchement du déclencheur et dans l'attribut liste de scripts disponibles.

8.1.6.3.2 Liste de paramètres d'exécution par défaut

Il s'agit d'une liste d'ids de script et de paramètres de scripts qui sont utilisés pour l'exécution par défaut lorsqu'un pas de lancement est déclenché sans paramètres.

8.1.6.3.3 Etat administratif

Cet attribut représente la capacité administrative d'un pas de lancement à remplir sa fonction. Les états administratifs suivants sont définis:

a) déverrouillé – Le pas de lancement est autorisé à démarrer ou à reprendre l'exécution des scripts de lancement;

b) verrouillé – Le pas de lancement n'est pas autorisé à démarrer l'exécution des pas de lancement. Si des exécutions sont en cours, elles sont suspendues.

8.1.6.3.4 Etat opérationnel

Cet attribut représente la capacité opérationnelle du pas de lancement à remplir ses fonctions.

Les états opérationnels suivants sont définis:

a) activé – Le pas de lancement est opérationnel et est prêt à exécuter un script;

b) désactivé – Le pas de lancement n'est pas opérationnel et n'est pas disponible pour des exécutions de script.

8.1.6.3.5 Etat d’utilisation

Cet attribut représente l'état d'utilisation du pas de lancement. Les états d'utilisation suivants sont définis:

a) occupé – Le pas de lancement est en cours d'utilisation pour l'exécution d'un script de lancement;

b) au repos – Le pas de lancement n'est pas en cours d'utilisation pour l'exécution d'un script.

8.1.6.3.6 Etat de disponibilité

Cet état indique que le pas de lancement est disponible pour remplir sa fonction. Les états suivants sont définis:

a) hors service

Le pas de lancement a été désactivé par un processus de commande interne conformément à un calendrier prédéterminé;

b) non hors service

L'attribut état de disponibilité ne porte pas la valeur hors service et, par conséquent, le pas de lancement a été activé.

8.1.6.3.7 Instance d'objet observé

Cette instance est définie dans la Rec. UIT-T X.739 | ISO/CEI 10164-11.

8.1.6.3.8 Id d'attribut observé

Cet id est défini dans la Rec. UIT-T X.739 | ISO/CEI 10164-11.

8.1.6.3.9 Etat de commande

Cet attribut est défini dans la Rec. X.731 du CCITT | ISO/CEI 10164-2. La valeur par défaut est vide. Si l'exécution est suspendue, l'état de commande passe à "suspendu" et si l'exécution est reprise, la valeur passe à vide.

8.1.6.3.10 Id de pas de lancement

Cet attribut désigne une instance de la classe d'objets gérés pas de lancement.

8.1.6.4 Notifications de la classe d'objets gérés pas de lancement

La classe d'objets gérés pas de lancement dispose des notifications suivantes qui sont transmises aux destinations de notification appropriées:

- résultat de déclenchement comme défini au 8.2.1;
- alarme d'erreur de traitement comme défini dans la Rec. X.733 du CCITT | ISO/CEI 10164-4.

8.1.7 Pas de lancement asynchrone

8.1.7.1 Aperçu général

- Sous-classe de la classe d'objets gérés pas de lancement.
- Des tâches élémentaires interruptibles sont générées de manière dynamique par un pas de lancement asynchrone.
- Transmet une notification de résultat de déclenchement dès que toutes les tâches élémentaires interruptibles ont été générées.

8.1.7.2 Blocs de la classe d'objets gérés pas de lancement asynchrone

8.1.7.3 Classe d'objets gérés

La classe d'objets gérés pas de lancement asynchrone contient le bloc suivant:

- triggerAsynchronousResultPackage (bloc résultat de déclenchement asynchrone).

8.1.8 Pas de lancement synchrone

8.1.8.1 Aperçu général

- Sous-classe de la classe d'objets gérés pas de lancement.
- Des tâches élémentaires sont générées de manière dynamique par des pas de lancement synchrones.
- Transmet une notification de résultat de déclenchement après les résultats de synchronisation dès que toutes les tâches élémentaires sont terminées.

8.1.8.2 Blocs de la classe d'objets gérés pas de lancement synchrone

La classe d'objets gérés pas de lancement synchrone contient le bloc suivant:

- bloc résultat de déclenchement synchrone.

8.1.9 Script de chaîne générale

8.1.9.1 Aperçu général

- Sous-classe de la classe d'objets script de lancement.
- Scripts qui peuvent être représentés sous la forme d'une chaîne générale.
- Il est possible d'ajouter, en tant que sous-classes, d'autres scripts dédiés.

8.1.9.2 Caractéristiques du langage de description de chaîne générale

La classe d'objets gérés langage de description de chaîne générale dispose des attributs suivants.

8.1.9.2.1 Nom de langage de description

Il s'agit du nom de langage qui définit les caractéristiques syntactiques et sémantiques d'un script qui est représenté par une chaîne générale.

8.1.9.2.2 Contenu du script

Ceci désigne le script qui est représenté comme une chaîne générale.

8.1.9.3 Blocs du langage de description de chaîne générale

La classe d'objets gérés langage de description de chaîne générale contient les blocs obligatoires suivants:

- bloc script de chaîne générale.

8.1.10 Référenceur de script

8.1.10.1 Aperçu général

- Hyperclasse pour les classes d'objets gérés pas de lancement et tâche élémentaire.
- Définit un mappage de rapport de référence entre pas de lancement et script de lancement, d'une part, et entre tâche élémentaire et script de lancement, d'autre part.

8.1.10.2 Blocs du référenceur de script

Le référenceur de script contient le bloc obligatoire suivant:

- bloc référenceur de script.

8.2 Notifications génériques

Les notifications suivantes sont définies dans la présente Spécification.

8.2.1 Résultat de déclenchement

Cette notification renvoie le résultat d'une exécution de script avec la valeur errorCode (code d'erreur) positionnée sur noError (pas d'erreur) si l'exécution a réussi ou un code d'erreur approprié pour indiquer la nature de la défaillance. Les codes d'erreur qui peuvent être renvoyés dans le champ errorCode de la notification executionResult (résultat d'exécution) sont les suivants:

- pas d'erreur si l'exécution a réussi;
- pas d'erreur de script si l'exécution a échoué parce que le nom du script n'était pas spécifié;
- erreur script rejeté s'il n'y avait pas de script dans la liste de scripts configurés pour exécution par un pas de lancement donné;
- erreur type de paramètre invalide s'il y a non-concordance entre le type de paramètre attendu par le script et celui fourni par le script;
- erreur de valeur de paramètre non valide si la valeur fournie dans le paramètre est non valide, par exemple, hors plage;
- erreur de syntaxe du script si l'exécution du script a échoué du fait d'une erreur de syntaxe dans le script;
- erreur échec d'exécution de script si l'exécution du script a échoué pour des raisons autres qu'une syntaxe incorrecte;
- nombre de paramètres non valides si le nombre de paramètres fourni n'est pas conforme au nombre de paramètres attendu par le script;
- erreur accès non autorisé si l'accès à une ou à plusieurs instances d'objet à utiliser par le script est refusé.

8.2.2 Résultat d'exécution

Cette notification renvoie le résultat d'une exécution de script avec la valeur errorCode positionnée sur noError si l'exécution a réussi ou un code d'erreur approprié pour indiquer la nature de la défaillance. Les codes d'erreur qui peuvent être renvoyés dans le champ errorCode de la notification executionResult sont:

- pas d'erreur si l'exécution a réussi;
- pas d'erreur de script si l'exécution a échoué parce que le nom du script n'était pas spécifié;
- erreur script rejeté s'il n'y avait pas de script dans la liste de scripts configurés pour l'exécution par un pas de lancement donné;
- erreur de type de paramètre invalide s'il y a non-concordance entre le type de paramètre attendu par le script et celui fourni par le script;
- erreur de valeur de paramètre non valide si la valeur fournie dans le paramètre est non valide, par exemple, hors plage;
- erreur de syntaxe de script si l'exécution du script a échoué du fait d'une erreur de syntaxe dans le script;
- erreur d'échec d'exécution de script si l'exécution du script a échoué pour des raisons autres qu'une syntaxe incorrecte;
- nombre de paramètres non valides si le nombre de paramètres fourni n'est pas conforme au nombre de paramètres attendu par le script;
- erreur accès non autorisé si l'accès à une ou à plusieurs instances d'objet à utiliser par le script est refusé.

8.3 Actions génériques

Les types d'action suivants sont définis dans la présente Spécification. Ces actions ont été définies pour le pas de lancement et les classes d'objets gérés tâche élémentaire interruptible dans la Spécification.

8.3.1 Action suspendre

L'action suspendre transmise à un objet géré entraîne la suspension de l'exécution de script en cours d'exécution par l'objet. Les paramètres contenus dans l'action suspendre sont l'id de déclencheur permettant d'identifier cette action et soit l'id de tâche élémentaire si l'action est destinée à une tâche élémentaire, soit l'id de pas de lancement, si l'action est destinée à un pas de lancement.

8.3.2 Action reprendre

L'action reprendre transmise à un objet géré entraîne la reprise de l'exécution de script par l'objet lorsque cette exécution a été suspendue par une action suspendre précédente. Les paramètres contenus dans l'action suspendre sont l'id de déclencheur permettant d'identifier cette action et soit l'id de tâche élémentaire si l'action est destinée à une tâche élémentaire, soit l'id de pas de lancement, si l'action est destinée à un pas de lancement.

8.3.3 Action terminer

L'action terminer transmise à un objet géré entraîne l'arrêt inconditionnel de toute exécution de script par cet objet et tous les objets générés par cet objet. Le paramètre id de déclencheur est utilisé pour identifier cette action.

8.3.4 Action déclencher

Une action déclencher entraîne le démarrage de l'exécution de script. Cette action fournit le paramètre id de déclencheur, l'identifie et sa composante liste de paramètres d'exécution est constituée d'une séquence d'ids de script qui indique les scripts qu'il convient d'exécuter ainsi que les paramètres de script qui sont nécessaires pour l'exécution desdits scripts.

9 Services

9.1 Introduction

Le séquenceur de commandes fournit des services permettant de modifier les opérations du séquenceur de commandes et des scripts de lancement. En particulier, les opérations qui peuvent être appliquées à chaque instance d'un séquenceur de commandes et des scripts de lancement sont les suivantes:

- création d'instances de séquenceur de commandes, de pas de lancement et de script de lancement;
- suppression d'instances de pas de lancement et de script de lancement;
- modification d'attributs de séquenceur de commandes, de pas de lancement et de script de lancement;
- récupération des attributs de séquenceur de commandes, de pas de lancement et de script de lancement.

Outre les services ci-dessous de modifications des instances, cette fonction fournit des services de notification et des actions permettant de déclencher l'exécution de commandes.

9.2 Services de démarrage, d'arrêt, de modification et de récupération

Les services PT-CREATE, PT-DELETE, PT-SET et PT-GET peuvent être utilisés pour créer, supprimer, modifier et récupérer des valeurs d'attributs d'instances d'objet support de gestion de séquenceur de commandes et d'objets gérés script de lancement et pas de lancement.

9.3 Services de notification

9.3.1 Définition du service résultat d'exécution

Ce paragraphe prescrit le service de signalisation executionResultInfo (information de résultat d'exécution) tel que défini dans la présente Recommandation | Norme internationale et est mappé avec le service CMIS M-EVENT-REPORT.

Tableau 6 – Paramètres de signalisation de résultat d'exécution

Parameter name	Req/Ind	Rsp/Cnf
Invoke identifier	P	P
Mode	P	–
Managed object class	P	P
Managed object instance	P	P
Event type	M	C (=)
Event time	P	–
Event information		
trigger id	M	–
script id	M	–
thread id	M	–
error code	U	–
execution result type	U	–
execution result	U	–
Current time	–	P
Event reply	–	P
Errors	–	P

Tableau 7 – Paramètres de signalisation de résultat de déclenchement

Parameter name	Req/Ind	Rsp/Cnf
Invoke identifier	P	P
Mode	P	–
Managed object class	P	P
Managed object instance	P	P
Event type	M	C (=)
Event time	P	–
Event information		
trigger id	M	–
script id	U	–
thread id	–	–
script parameters	U	–
execution type	U	–
error code	U	–
execution result	U	–
Current time	–	P
Event reply	–	P
Errors	–	P

9.4 Services action

Ce paragraphe prescrit les services d'actions "déclencher" et "terminer" tels que définis dans la présente Recommandation | Norme internationale et est mappé avec le service CMIS M-EVENT-ACTION.

Tableau 8 – Paramètres du service d'action "déclencher"

Parameter name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access control	P	–
Synchronization	P	–
Action type	M	–
Action Information		
trigger id	M	–
Attribute list	U	–
Errors	–	P

Tableau 9 – Paramètres du service d'action "terminer"

Parameter name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access control	P	–
Synchronization	P	–
Action type	M	–
Action Information		
trigger id	M	–
Errors	–	P

Tableau 10 – Paramètres du service d'action "suspendre"

Parameter name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access control	P	–
Synchronization	P	–
Action type	M	–
Action Information		
trigger id	M	–
thread id	U	–
launch pad id	U	–
Errors	–	P

Tableau 11 – Paramètres du service de l'action "reprendre"

Parameter name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access control	P	–
Synchronization	P	–
Action type	M	–
Action Information		
trigger id	M	–
thread id	U	–
launch pad id	U	–
Errors	–	P

10 Unités fonctionnelles

La présente Recommandation | Norme internationale définit trois unités fonctionnelles pour la gestion des séquenceurs de commandes:

a) *unité fonctionnelle d'exécution*

L'unité fonctionnelle d'exécution nécessite les services de PT-CREATE, le service de notification de résultat d'exécution d'action de déclenchement et le service de signalisation d'alarme d'erreur de traitement;

b) *unité fonctionnelle de surveillance*

L'unité fonctionnelle de surveillance nécessite les services de PT-GET;

c) *unité fonctionnelle de commande*

L'unité fonctionnelle de commande nécessite les services de l'action "terminer", PT-DELETE.

11 Protocoles et syntaxe abstraite

11.1 Syntaxe abstraite

11.1.1 Objets gérés

11.1.1.1 Objets gérés définis

Le Tableau 12 indique la relation entre les objets gérés définis au 8.1 et les spécifications de la classe d'objets gérés définies dans l'Annexe A.

Tableau 12 – Objets gérés et étiquettes de référence

Managed object name	Reference label
Basic spawner	basicSpawnerClass
Suspendable thread	suspendableThread
Thread	thread
Asynchronous launch pad	asynchronousLaunchPad
Synchronous launch pad	synchronousLaunchPad
Launch pad	launchPad
Launch script	launchScript
Command sequencer	commandSequencer
General string script	generalStringScript
Script Referencer	scriptReferencer

11.2 Attributs

11.2.1 Attributs tirés de la définition des informations de gestion

La présente Spécification référence les attributs de gestion ci-dessous, dont la syntaxe abstraite est spécifiée dans la Rec. X.721 du CCITT | ISO/CEI 10165-2:

a) administrativeState;

b) usageState;

c) operationalState;

d) controlStatus;

e) availabilityStatus.

Elle référence également les attributs de gestion ci-dessous dont la syntaxe abstraite est spécifiée dans la Rec. UIT-T X.739 | ISO/CEI 10164-11:

a) observedObjectInstance;

b) observedAttributeId.

11.2.2 Attributs définis dans la présente Spécification

La présente Spécification définit les attributs de gestion suivants dont la syntaxe abstraite est spécifiée dans l'Annexe A:

a) launchPadId;

b) commandSequencerId;

c) threadId;

d) scriptId;

e) triggerId;

f) executionResultType;

g) executingParameters;

h) scriptLanguageName;

i) scriptContent;

j) defaultExecutionParameterList;

k) availableScriptList.

11.2.3 Mappage des paramètres et attributs

Le Tableau 13 identifie la relation entre les paramètres de service définis aux 8.1 et 8.2 et les spécifications des types d'attributs décrites dans l'Annexe A.

Tableau 13 – Paramètres et noms d'attributs

Parameter name	Attribute name
Launch pad id	launchPadId
Command sequencer id	commandSequencerId
Thread id	threadId
Script id	scriptId
Trigger id	triggerId
Execution result type	executionResultType
Executing parameters	executingParameters
Script language name	scriptLanguageName
Script content	scriptContent
Default execution parameter list	defaultExecutionParameterList
Available script list	availableScriptList

11.4 Notifications

11.4.1 Notifications référencées

La présente Spécification référence les événements suivants définis dans la Rec. X.730 du CCITT | ISO/CEI 10164-1:

a) notification de création d'objets;

b) notification de suppression d'objets;

c) notification d'alarme d'erreur de traitement.

La présente Spécification référence également l'événement suivant défini dans la Rec. X.730 du CCITT | ISO/CEI 10164-2:

– notification de changement d'état.

11.4.2 Notifications définies dans la présente Spécification

Le Tableau 14 recense la relation entre les notifications définies au 9.3 et les spécifications des types de notification décrites dans l'Annexe A.

Tableau 14 – Notifications

Event type	Notification type
Trigger result	triggerResultInfo
Execution result	executionResultInfo

11.5 Actions

11.5.1 Actions définies dans la présente Spécification

Le Tableau 15 recense la relation entre les actions définies au 9.4 et les spécifications du type de notification décrites dans l'Annexe A.

Tableau 15 – Actions

Action name	Reference label
terminate	terminate
suspend	suspend
resume	resume
trigger	trigger

11.6 Négociation des unités fonctionnelles

La présente Recommandation | Norme internationale assigne la valeur suivante d'identificateur d'objet:

{joint-iso-itu-t ms(9) function(2) part21(21) functionalUnitPackage(1)}

comme valeur FunctionalUnitPackageId type ASN.1 défini dans la Rec. X.701 du CCITT | ISO/CEI 10040 à utiliser pour négocier les unités fonctionnelles suivantes:

0 Unité fonctionnelle d'exécution

M Unité fonctionnelle de surveillance

2 Unité fonctionnelle de commande

dans lesquelles le nombre désigne les positions des bits dans les CHAÎNES BINAIRES affectées aux unités fonctionnelles et les noms de référence des unités fonctionnelles sont définis à l'article 10.

Dans le contexte d'application de gestion-systèmes, le mécanisme pour négocier les unités fonctionnelles est décrit dans la Rec. X.701 du CCITT | ISO/CEI 10040.

NOTE – C'est le contexte d'application qui dicte la nécessité de négocier les unités fonctionnelles.

12 Relations avec d'autres fonctions

Le séquenceur de commandes utilise les services définis dans la Rec. X.731 du CCITT | ISO/CEI 10164-2 pour la notification des changements d'état; il utilise les services définis dans la Rec. X.730 du CCITT | ISO/CEI 10164-1 pour la création et la suppression d'objets gérés, la récupération d'attributs et la notification des changements de valeur d'attributs.

Le séquenceur de commandes utilise les services définis dans la Rec. UIT-T X.741 | ISO/CEI 10164-9 pour la fourniture de capacités de commande d'accès aux instances d'objets gérés qui peuvent être exploitées par les tâches élémentaires.

13 Conformité

Il existe deux classes de conformité: la classe de conformité générale et la classe de conformité induite. Un système qui prétend mettre en œuvre les éléments de procédure pour les services de gestion-systèmes définis dans la présente Spécification doit satisfaire aux spécifications de la classe de conformité générale, ou de conformité induite définies ci-après. Le fournisseur de la mise en œuvre doit indiquer la classe.

13.1 Spécifications de la classe de conformité générale

Un système qui prétend à une conformité générale doit prendre en charge la présente fonction pour toutes les classes d'objets gérés qui importent l'information de gestion définie dans la présente Spécification.

NOTE – Cette spécification est applicable à toutes les sous-classes de classes d'objet support de gestion définies dans la présente Spécification.

13.1.1 Conformité statique

Le système doit:

a) prendre en charge le rôle de gestionnaire ou d'agent ou les deux vis-à-vis de l'unité fonctionnelle de commande et de l'unité fonctionnelle de surveillance;

b) prendre en charge la syntaxe de transfert obtenue des règles de codage spécifiées dans la Rec. X.209 du CCITT | ISO/CEI 8825 et appelée {joint-iso-itu-t asn1(1) basicEncoding(1)}, pour générer et interpréter les MAPDU définies par les types de données abstraites référencés aux 11.4 et 11.5;

c) lorsqu'il agit en tant qu'agent, prendre en charge une ou plusieurs instances d'au moins une des classes d'objets gérés séquenceur de commandes, pas de lancement, script de lancement, tâche élémentaire ou l'une de leurs sous-classes.

13.1.2 Conformité dynamique

Le système doit, dans le ou les rôles pour lesquels une conformité est prétendue:

a) prendre en charge les éléments de procédure définis dans:

- la Rec. X.730 du CCITT | ISO/CEI 10164-1 pour les services PT-GET, PT-CREATE, PT-DELETE, PT-SET, signalisation de création d'objets, de suppression d'objet et de modification d'attribut;
- la Rec. X.731 du CCITT | ISO/CEI 10164-2 pour le service de signalisation de changement d'état;
- la Rec. X.733 du CCITT | ISO/CEI 10164-4 pour le service de signalisation d'alarme d'erreur de traitement;

b) prendre en charge les éléments de procédure définis dans la présente Spécification pour les services de signalisation et d'action suivants:

- notification de résultat d'exécution;
- action déclencher;
- action terminer;
- action suspendre;
- action reprendre.

13.2 Spécifications de la classe de conformité induite

13.2.1 Conformité statique

Le système doit:

a) prendre en charge la syntaxe de transfert obtenue des règles de codage spécifiées dans la Rec. X.209 du CCITT | ISO/CEI 8825 et appelée {joint-iso-itu-t asn1(1) basicEncoding(1)}, pour générer et interpréter les MAPDU définies par les types de données abstraites référencés au 11.1, comme l'exige une spécification de référence;

b) prendre en charge une ou plusieurs instances d'une des classes d'objets gérés séquenceur de commandes, pas de lancement, script de lancement, tâche élémentaire ou l'une de leurs sous-classes, lorsqu'il joue le rôle d'agent.

13.2.2 Conformité dynamique

Le système doit prendre en charge les éléments de procédure référencés dans la présente Spécification comme l'exige une spécification de référence.

13.3 Conformité pour la prise en charge des définitions d'objets gérés

Les objets du séquenceur de commandes pris en charge par le système ouvert doivent satisfaire au comportement spécifié dans l'article 8 et à la syntaxe prescrite en Annexe A.

Annexe A

Définition des informations de gestion

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

A.1 Définitions des classes d'objets gérés

A.1.1 Objets de base

```
basicSpawnerClass   MANAGED OBJECT CLASS
DERIVED FROM "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":top;
CHARACTERIZED BY basicSpawnerPackage ;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx1(1)};


commandSequencer    MANAGED OBJECT CLASS
DERIVED FROM "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":top;
CHARACTERIZED BY
       commandSequencerPackage   PACKAGE
       BEHAVIOUR commandSequencerBehaviour BEHAVIOUR
       DEFINED AS "Une instance de cette classe représente une ressource jouant un rôle de
       gestionnaire en tant qu'invocateur d'opérations déterminées par ses scripts de
       lancement.";;
       ATTRIBUTES
              commandSequencerId    GET,
       "Rec. X.731 du CCITT |ISO/CEI 10164-2:1992": administrativeState GET-REPLACE,
              "Rec. X.731 du CCITT |ISO/CEI 10164-2:1992":operationalState GET;
       NOTIFICATIONS
              "Rec. X.730 du CCITT | ISO/CEI 10164-1": objectCreation,
              "Rec. X.730 du CCITT | ISO/CEI 10164-1": objectDeletion,
              "Rec. X.731 du CCITT | ISO/CEI 10164-2": stateChange;;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21)
managedObjectClass(3) xx2(2)};

generalStringScript MANAGED OBJECT CLASS
DERIVED FROM launchScript;
CHARACTERIZED BY generalStringScriptPackage;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx3(3)};

asynchronousLaunchPad MANAGED OBJECT CLASS
DERIVED FROM launchPad;
CHARACTERIZED BY triggerAsynchronousResultPackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx4(4)};

synchronousLaunchPad MANAGED OBJECT CLASS
DERIVED FROM launchPad;
CHARACTERIZED BY triggerSynchronousResultPackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx5(5)};
```

```
launchPad    MANAGED OBJECT CLASS
DERIVED FROM basicSpawnerClass, scriptReferencer;
CHARACTERIZED BY
           launchPadPackage,
             triggerActionAccepter,
             parameterPasser,
             triggerResultPackage,
             triggerEventAccepter,
             terminateAccepter,
             "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": externalScheduler,
             suspendResumeAccepter;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx6(6)};

launchScript MANAGED OBJECT CLASS
DERIVED FROM "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":top;
CHARACTERIZED BY
       launchScriptPackage PACKAGE
       BEHAVIOUR launchScriptBehaviour BEHAVIOUR
       DEFINED AS "Cet objet géré représente les instructions à exécuter par un séquenceur
       de commandes.";;
       ATTRIBUTES
              scriptId GET,
              executionResultType GET,
              "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": administrativeState
              GET-REPLACE;;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx7(7)};

--
-- L'hyperclasse non instanciable suivante simplifie la description de la relation entre
-- un pas de lancement et ses scripts ainsi que la description de la relation entre
-- les tâches élémentaires et les scripts. Elle est comprise dans les hiérarchies
-- d'héritage des classes pas de lancement et des tâches élémentaires.
--
scriptReferencer MANAGED OBJECT CLASS
DERIVED FROM "Rec. UIT-T X.725 | ISO/CEI 10165-7": genericRelationshipObject;
CHARACTERIZED BY scriptReferencerPackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx8(8)};

thread MANAGED OBJECT CLASS
       DERIVED FROM basicSpawnerClass, scriptReferencer;
       CHARACTERIZED BY threadPackage, executionResultPackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx9(9)};

suspendableThread MANAGED OBJECT CLASS
       DERIVED FROM thread;
       CHARACTERIZED BY suspendResumeAccepter;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx10(10)};
```

A.2 Définitions des blocs

A.2.1 Blocs de base

```
basicSpawnerPackage  PACKAGE
        BEHAVIOUR spawnerBehaviour;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx1(1)};

generalStringScriptPackage PACKAGE
        BEHAVIOUR generalStringScriptBehaviour;
        ATTRIBUTES scriptLanguageName GET-REPLACE,
                scriptContent GET-REPLACE;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx2(2)};

parameterPasser PACKAGE
        BEHAVIOUR
        parameterPasserBehaviour;
        ATTRIBUTES
            "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": administrativeState,
            "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": operationalState,
            "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": usageState,
            "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": availabilityStatus;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx3(3)};

executionResultPackage   PACKAGE
        BEHAVIOUR executionResultBehaviour;;
        NOTIFICATION executionResultInfo;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx4(4)};

launchPadPackage PACKAGE
        BEHAVIOUR launchPadBehaviour;
        ATTRIBUTES launchPadId GET;
        NOTIFICATIONS
        "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": processingErrorAlarm;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx5(5)};

scriptReferencerPackage PACKAGE
        BEHAVIOUR scriptReferencerBehaviour;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx6(6)};

suspendResumeAccepter PACKAGE
        BEHAVIOUR suspendResumeBehaviour;
        ATTRIBUTES "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":controlStatus GET;
        ACTIONS suspend, resume;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx7(7)};
```

```
terminateAccepter PACKAGE
       ACTIONS terminate;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx8(8)};

threadPackage PACKAGE
        BEHAVIOUR threadBehaviour,
                   simpleScriptExecutionBehaviour;
       ATTRIBUTES scriptId GET,
                  threadId GET,
                  executingParameters   GET SET-BY-CREATE,
                       "Rec. X.731 du CCITT |ISO/CEI 10164-2:1992": operationalState GET;
        NOTIFICATIONS
               "Rec. X.734 du CCITT |ISO/CEI 10164-5": processingErrorAlarm;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx9(9)};

triggerActionAccepter PACKAGE
        BEHAVIOUR   spawnerBehaviour,
                      triggerActionAccepterBehaviour;;
        ATTRIBUTES defaultExecutionParameterList REPLACE WITH DEFAULT
                   GET-REPLACE
                   SET BY CREATE
                   DEFAULT VALUE CSModule.emptyExecutionParameterList;
                availableScriptList REPLACE WITH DEFAULT
                   ADD-REMOVE
                   GET-REPLACE
                   SET BY CREATE
                   DEFAULT VALUE CSModule.emptyScriptList;
        ACTIONS Trigger;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx10(10)};

triggerEventAccepter PACKAGE
        BEHAVIOUR triggerEventAccepterBehaviour;
        ATTRIBUTES
           "Rec. UIT-T X.739 (1993)|ISO/CEI 10164-11:1994": observedObjectInstance
GET-REPLACE,
           "Rec. UIT-T X.739 (1993)|ISO/CEI 10164-11:1994": observedAttributeId
GET-REPLACE;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx11(11)};

triggerAsynchronousResultPackage PACKAGE
        BEHAVIOUR triggerAsynchronousResultBehaviour;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx12(12)};

triggerSynchronousResultPackage PACKAGE
        BEHAVIOUR triggerSynchronousResultBehaviour;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx13(13)};
```

```
triggerResultPackage PACKAGE

        BEHAVIOUR    triggerResultBehaviour;;

        NOTIFICATION triggerResultInfo;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx14(14)};
```

A.3 Définitions des comportements

```
spawnerBehaviour  BEHAVIOUR

       DEFINED AS ! Les instances de cette classe sont capables de donner lieu à la
       création de nouvelles instances d'objets. Les instances nouvellement créées seront
       contenues par cette instance et leurs noms seront automatiquement générés. Jusqu'à
       ce que tous les objets créés soient complets, l'état d'utilisation de l'objet sera
       "en cours d'utilisation". Si l'état administratif de cet objet est "verrouillé",
       les nouveaux objets ne peuvent être créés. Si, du fait d'une limitation des
       ressources locales, cet objet est incapable de prendre en charge un plus grand
       nombre d'objets contenus, son état d'utilisation sera "occupé".



       Lorsqu'une instance de cette classe donne lieu à la création de nouveaux objets,
       elle est utilisée comme un objet IVMO pendant la création en fournissant les
       valeurs d'attributs de nouvel objet sur la base de son propre attribut
       defaultExecutionParameterList ou d'éventuels paramètres qui lui ont été fournis
       dans le cadre de l'action du mécanisme local qui a déclenché la génération de la
       nouvelle instance.



       Une instance de cette classe peut donner lieu à la création de nouvelles instances
       d'objets à partir d'un seul id de script, d'un ensemble d'ids de script dans
       n'importe quel ordre ou d'une séquence d'ids de script dans l'ordre spécifié dans
       la liste, c'est-à-dire après création de la première instance, la seconde ne peut
       être créée qu'une fois la première terminée etc. La valeur de l'attribut scriptId
       de l'objet créé obtient sa valeur de l'élément correspondant de la liste de scripts
       de cet objet.!;



executionResultBehaviour BEHAVIOUR

DEFINED AS "Les instances d'une classe qui prend en charge ce comportement signalent les
résultats intermédiaires et finaux de l'exécution d'une tâche élémentaire.";



triggerAsynchronousResultBehaviour BEHAVIOUR

       DEFINED AS "Dès lancement de toutes les tâches élémentaires qui doivent être
       lancées par un déclencheur, le pas de lancement émet la notification
       triggerResultInfo.";



triggerSynchronousResultBehaviour BEHAVIOUR

       DEFINED AS "Dès achèvement de toutes les tâches élémentaires qui doivent être
       lancées par un déclencheur, le pas de lancement émet la notification
       triggerResultInfo qui contient des résultats d'exécution ou des erreurs.";



triggerResultBehaviour BEHAVIOUR

       DEFINED AS "Le pas de lancement émet la notification triggerResultInfo.";



scriptReferencerBehaviour BEHAVIOUR

       DEFINED AS "Un référenceur de script est une classe d'objets non instanciable qui
       définit une mise en correspondance de relations de références à partir des
       instances de classes d'objets gérés script de lancement et pas de lancement et à
       partir d'instances de classes d'objets gérés script de lancement et tâche
       élémentaire.";
```

```
suspendResumeBehaviour BEHAVIOUR

      DEFINED AS "L'exécution d'un script par une tâche élémentaire peut être suspendue
      par une action "suspendre" envoyée à la tâche élémentaire ou au pas de lancement
      et, par la suite, remise en service par une action reprendre. La valeur par défaut
      de controlStatus est vide. Si l'action suspendre est exécutée, la valeur change en
      "suspendu". Après exécution de l'action reprendre, la valeur revient à vide.";


triggerEventAccepterBehaviour BEHAVIOUR

      DEFINED AS "Le pas de lancement a des attributs permettant de surveiller un
      attribut spécifique dans une instance d'objet particulière. Si la valeur de
      l'attribut surveillé est modifiée, il est généré un déclencheur pour lancer les
      scripts spécifiés par les ids de scripts indiqués dans l'attribut liste des
      paramètres d'exécution par défaut. Dans le cas où l'attribut surveillé est un
      compteur EDC (compteur de discrimination d'événement) tel que défini en Annexe C,
      les notifications déclenchent, par l'intermédiaire d'un EDC, le lancement de
      l'exécution de scripts par le pas de lancement.";


triggerActionAccepterBehaviour BEHAVIOUR

      DEFINED AS "Lorsqu'une instance de cette classe est en service et qu'elle reçoit un
      déclenchement, à partir d'un activateur de déclenchement, si son attribut scriptId
      n'est pas vide, un nouvel objet est créé dans lequel l'id de script et la classe de
      la nouvelle instance proviennent de la valeur de l'attribut scriptId de cette
      instance et tous les éventuels autres attributs et paramètres transmis par le
      déclencheur.";


threadBehaviour BEHAVIOUR

      DEFINED AS "Lorsqu'une instance d'un objet de cette classe est créée, elle commence
      l'exécution de la séquence de commande spécifiée dans ses attributs, au moyen de sa
      liste de paramètres afin de fournir tout paramètre nécessaire au script. Lorsque
      l'exécution de cette séquence est terminée, l'objet est supprimé. Si l'exécution du
      script donne lieu à la création de tâches élémentaires contenues, cette tâche
      élémentaire n'est considérée terminée qu'une fois toutes les tâches élémentaires
      contenues terminées.";


parameterPasserBehaviour BEHAVIOUR

      DEFINED AS "Une instance d'objet de cette classe transmet un ensemble de paramètres
      à une instance d'objet d'une autre classe.";


simpleScriptExecutionBehaviour BEHAVIOUR
      DEFINED AS "Un script est exécuté ou interprété par des ressources locales. Son
      état d'exécution reflète les états suivants:

            – activé (transition spontanée à l'état suivant: en cours d'exécution);

            – en cours d'exécution (état suivant: temporisation ou exécution terminée);

            – temporisation (transition spontanée à l'état suivant: terminé);

            – terminé.

       NOTE – La valeur de la temporisation dépend de l'application particulière.";


generalStringScriptBehaviour BEHAVIOUR

      DEFINED AS "La syntaxe et la sémantique du langage de description qui peuvent être
      représentées comme une chaîne générale. Se reporter aux Annexes F et G pour plus de
      détails sur le langage, SMSL (langage de description pour la gestion-systèmes).";
```

A.4 Définitions des attributs

```
availableScriptList  ATTRIBUTE

       WITH ATTRIBUTE SYNTAX CSModule.AvailableScriptList;

       MATCHES FOR EQUALITY;

       BEHAVIOUR
```

```
        availableScriptListBehaviour BEHAVIOUR
        DEFINED AS "Un ensemble de noms d'instances d'objets gérés des instructions de
        script qui peuvent être exécutées par un pas de lancement.";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx1(1)};


commandSequencerId  ATTRIBUTE
       WITH ATTRIBUTE SYNTAX CSModule.CommandSequencerId;
       MATCHES FOR EQUALITY;
       BEHAVIOUR
       commandSequencerIdBehaviour   BEHAVIOUR
       DEFINED AS "Le nom d'instance d'objets gérés du séquenceur de commandes.";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx2(2)};


executionResultType ATTRIBUTE
        WITH ATTRIBUTE SYNTAX CSModule.ExecutionResultType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
        executionResultTypeBehaviour BEHAVIOUR
        DEFINED AS "Ceci indique le type de résultat d'exécution.";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)};


scriptContent ATTRIBUTE
        WITH ATTRIBUTE SYNTAX CSModule.ScriptContent;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
        scriptContentBehaviour BEHAVIOUR
        DEFINED AS "Le contenu d'un script de lancement représenté par une chaîne
        générale.";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx4(4)};


scriptId   ATTRIBUTE
       WITH ATTRIBUTE SYNTAX CSModule.ScriptId;
       MATCHES FOR EQUALITY;
       BEHAVIOUR
       scriptIdBehaviour   BEHAVIOUR
       DEFINED AS "Le nom de l'instance d'objets gérés du script à exécuter.";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)};


launchPadId  ATTRIBUTE
       WITH ATTRIBUTE SYNTAX CSModule.LaunchPadId;
       MATCHES FOR EQUALITY;
       BEHAVIOUR
       launchPadIdBehaviour   BEHAVIOUR
       DEFINED AS "Le nom de l'instance d'objets gérés du pas de lancement.";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx6(6)};
```

```
scriptLanguageName ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.ScriptLanguageName;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        scriptLanguageNameBehaviour BEHAVIOUR

        DEFINED AS "Le nom de l'instance d'objets gérés du script de lancement représenté
        par une chaîne générale.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx7(7)};


defaultExecutionParameterList  ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.ExecutionParameterList;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        defaultExecutionParameterListBehaviour BEHAVIOUR

        DEFINED AS "Un ensemble de noms d'instances d'objets gérés des instructions de
        script et des valeurs de paramètres (si nécessaire) utilisées comme entrées pour
        les instances à exécuter par défaut.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx8(8)};


executingParameters   ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.ExecutionParameter;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        executingParametersBehaviour BEHAVIOUR

        DEFINED AS "Un ensemble de noms d'instances d'objets gérés des instructions de
        script et des valeurs de paramètres (si nécessaire) utilisées comme entrées pour
        les exécutions de script.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx9(9)};


threadId   ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.ThreadId;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        threadIdBehaviour   BEHAVIOUR

        DEFINED AS "Le nom d'instance d'objets gérés d'une tâche élémentaire exécutant
        de(s) instruction(s) de script.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx10(10)};


triggerId ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.TriggerId;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        triggerIdBehaviour BEHAVIOUR

        DEFINED AS "Le nom d'instance d'objets gérés d'un déclencheur amorçant l'exécution
        d'un script de lancement.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx11(11)};
```

A.5 Définitions des notifications

```
executionResultInfo   NOTIFICATION

       WITH INFORMATION SYNTAX CSModule.ExecutionResultInfo;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) notification(10) xx1(1)};

triggerResultInfo NOTIFICATION

       WITH INFORMATION SYNTAX CSModule.TriggerResultInfo;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) notification(10) xx2(2)};
```

A.6 Définitions des actions

```
resume ACTION

       BEHAVIOUR resumeBehaviour BEHAVIOUR

             DEFINED AS "Une action destinée à un objet basicSpawnerClass entraînant une
             reprise inconditionnelle de toutes les exécutions de scripts par l'objet
             basicSpawnerClass lorsque ses exécutions ont été amorcées par un déclencheur
             particulier. La valeur de controlStatus devient vide en conséquence de
             l'action reprendre.";;

       WITH INFORMATION SYNTAX CSModule.SpawnerObjectId;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx1(1)};


suspend ACTION

       BEHAVIOUR suspendBehaviour BEHAVIOUR

       DEFINED AS "Une action destinée à un objet basicSpawnerClass entraînant la
       suspension inconditionnelle de toutes les exécutions de scripts par l'objet
       basicSpawnerClass lorsque ses exécutions ont été amorcées par un déclencheur
       particulier. La valeur de controlStatus devient "suspendue" en conséquence de
       l'action suspendre.";;

       WITH INFORMATION SYNTAX CSModule.SpawnerObjectId;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx2(2)};


terminate ACTION

       BEHAVIOUR terminateBehaviour BEHAVIOUR

DEFINED AS "Une action destinée à un pas de lancement, entraînant une terminaison
inconditionnelle de tous les scripts par le pas de lancement, lorsque ses scripts ont été
amorcés par un déclencheur particulier.";;

       WITH INFORMATION SYNTAX CSModule.TriggerId;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx3(3)};


trigger ACTION

       BEHAVIOUR triggerBehaviour BEHAVIOUR

       DEFINED AS "Un initiateur d'exécution de script en faisant en sorte qu'un pas de
       lancement génère de manière dynamique une ou plusieurs tâches élémentaires.";;

       WITH INFORMATION SYNTAX CSModule.TriggerParameters;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx4(4)};
```

A.7 Définitions des rattachements de noms

```
commandSequencer-system   NAME BINDING

       SUBORDINATE OBJECT CLASS commandSequencer AND SUBCLASSES;

       NAMED BY SUPERIOR OBJECT CLASS system AND SUBCLASSES;

       WITH ATTRIBUTE commandSequencerId;
```

```
        BEHAVIOUR csSystemContainmentBehaviour BEHAVIOUR

        DEFINED AS "La classe de l'objet supérieur est "system" et la classe de l'objet
        subordonné est commandSequencer.";;

        CREATE WITH-REFERENCE-OBJECT, WITH-AUTOMATIC-INSTANCE-NAMING;

        DELETE ONLY-IF-NO-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx1(1)};


launchPad-commandSequencer NAME BINDING

        SUBORDINATE OBJECT CLASS launchPad AND SUBCLASSES;

        NAMED BY SUPERIOR OBJECT CLASS commandSequencer AND SUBCLASSES;

        WITH ATTRIBUTE launchPadId;

        BEHAVIOUR lpCsContainmentBehaviour BEHAVIOUR

        DEFINED AS "Le fait de nommer un pas de lancement d'une séquence de commandes, par
        rapport au séquenceur de commandes indique que le séquenceur de commandes est le
        fournisseur de service pour le pas de lancement.";;

        CREATE WITH-AUTOMATIC-INSTANCE-NAMING;

        DELETE DELETES-CONTAINED-OBJECTS;

        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6)
xx2(2)};


thread-synchronousLaunchPad  NAME BINDING

        SUBORDINATE OBJECT CLASS   thread AND SUBCLASSES;

        NAMED BY SUPERIOR OBJECT CLASS synchronousLaunchPad AND SUBCLASSES;

        WITH ATTRIBUTE   threadId;

        BEHAVIOUR threadSyncLpContainmentBehaviour BEHAVIOUR

        DEFINED AS "La classe d'objets supérieurs synchronousLaunchPad agit comme un objet
        IVMO pour la classe d'objets subordonnés tâche élémentaire.";;

        DELETE DELETES-CONTAINED-OBJECTS;

        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21)

nameBinding(6) xx3(3)};


suspendableThread-asynchronousLaunchPad   NAME BINDING

        SUBORDINATE OBJECT CLASS   suspendableThread AND SUBCLASSES;

        NAMED BY SUPERIOR OBJECT CLASS asynchronousLaunchPad AND SUBCLASSES;

        WITH ATTRIBUTE   threadId;

        BEHAVIOUR threadAsyncLpContainmentBehaviour BEHAVIOUR

        DEFINED AS "La classe d'objets supérieurs asynchronousLaunchPad agit comme un
        objet IVMO pour la classe d'objets subordonnés tâche interruptible.";;

        DELETE DELETES-CONTAINED-OBJECTS;

        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21)

nameBinding(6) xx4(4)};


thread-thread   NAME BINDING

        SUBORDINATE OBJECT CLASS   thread AND SUBCLASSES;

        NAMED BY SUPERIOR OBJECT CLASS thread AND SUBCLASSES;

        WITH ATTRIBUTE threadId;
```

```
        BEHAVIOUR threadContainmentBehaviour BEHAVIOUR

        DEFINED AS "La classe d'objets supérieurs tâche élémentaire agit comme un
        générateur dynamique de la classe d'objets subordonnés tâche élémentaire.";;

        DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx5(5)};


suspendableThread-suspendableThread   NAME BINDING

        SUBORDINATE OBJECT CLASS   suspendableThread AND SUBCLASSES;

        NAMED BY SUPERIOR OBJECT CLASS suspendableThread AND SUBCLASSES;

        WITH ATTRIBUTE threadId;

        BEHAVIOUR suspendableThreadContainmentBehaviour BEHAVIOUR

        DEFINED AS "La classe d'objets supérieurs tâche interruptible agit comme un
        générateur dynamique de la classe d'objets subordonnés tâche élémentaire
        interruptible.";;

        DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx6(6)};


launchScript-system NAME BINDING

        SUBORDINATE OBJECT CLASS launchScript AND SUBCLASSES;

        NAMED BY SUPERIOR OBJECT CLASS system AND SUBCLASSES;

        WITH ATTRIBUTE scriptId;

        BEHAVIOUR lsSystemContainmentBehaviour BEHAVIOUR

        DEFINED AS "La classe d'objets supérieurs est "system" et la classe d'objets
        subordonnés est launchScript.";;

        CREATE WITH-REFERENCE-OBJECT, WITH-AUTOMATIC-INSTANCE-NAMING;

        DELETE ONLY-IF-NO-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx7(7)};
```

A.8 Définitions ASN.1

```
CSModule {joint-iso-itu-t ms(9) function(2) part21(21) asn1Module(2) 0}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN


-- EXPORTE tout


IMPORTS

        SimpleNameType

       FROM Attribute-ASN1Module {joint-iso-itu-t ms(9) smi(3) part2(2)
asn1Module(2) 1 }

       ObjectInstance, Attribute, CMISSync, CMISFilter, ModifyOperator, Scope,

       BaseManagedObjectId FROM CMIP-1 {joint-iso-itu-t ms(9) cmip(1) modules(0)
protocol(3)}

       AE-title FROM ACSE-1 {joint-iso-itu-t association-control(2) abstract-syntax(1)
apdus(0) version(1)};


       cmdSeqRelationshipClasses OBJECT IDENTIFIER ::= {joint-iso-itu-t ms(9) function(2)
part21(21) relationshipClass(11) }
```

```
      cmdSeqRelationshipMappings OBJECT IDENTIFIER ::= {joint-iso-itu-t ms(9) function(2)
part21(21) relationshipMapping(12)}

       cmdSeqRelationshipRoles OBJECT IDENTIFIER ::= {joint-iso-itu-t ms(9) function(2)
part21(21) relationshipRole(13)}


--
-- Contraintes de plage utilisées pour des définitions de classes de relations
--
RangeFromOneToOne ::= INTEGER (1 .. 1)
RangeFromZeroToMax ::= INTEGER (0 .. MAX)


--
-- Contrainte de taille de compteur
--
MaxCounterSize ::= INTEGER{unlimited(0)}-- size in octets


ExecutionResultInfo ::= SEQUENCE {triggerId TriggerId,
                                  scriptId ScriptId,
                                  threadId ThreadId,
                                  errorCode ErrorCode,
                                  executionResultType ExecutionResultType,
                                  executionResult SET OF Attribute}

TriggerResultInfo ::=  SEQUENCE {triggerId TriggerId,
                          CHOICE {singleTriggerResult ResultInfoFromThread,
                                  sequentialTriggerResult SEQUENCE OF
                                                      ResultInfoFromThread,
                                  parallelTriggerResult SET OF ResultInfoFromThread}}

ResultInfoFromThread ::= SEQUENCE{executionType  ExecutionType,
                                  errorCode      ErrorCode,
                                  executionResultType ExecutionResultType,
                                  executionResult SET OF Attribute}

ExecutionType ::= CHOICE {singleExecution ScriptThreadSet,
                          parallelExecution SET OF ScriptThreadSet,
                          sequentialExecution SEQUENCE OF ScriptThreadSet}

ScriptThreadSet ::= SEQUENCE {scriptId  ScriptId,
                              threadId    ThreadId}

SpawnerObjectId ::= SEQUENCE  {triggerId  TriggerId,
                               CHOICE { threadId ThreadId,
                                        launchPadId LaunchPadId}}
```

```
ExecutionResultType ::= OBJECT IDENTIFIER
CommandSequencerId ::=  ObjectInstance
ScriptId ::=  ObjectInstance
ThreadId ::=  ObjectInstance
TriggerId ::= ObjectInstance
LaunchPadId ::= ObjectInstance

ScriptList ::= CHOICE {scriptId ScriptId,
                       sequentialScriptList SEQUENCE OF ScriptId,
                       parallelScriptList SET OF ScriptId}
AvailableScriptList ::= SET OF ScriptList
emptyScriptList AvailableScriptList ::= {}
emptyExecutionParameterList ExecutionParameterList ::= sequentialExecutionList:{}

TriggerParameters  ::= SEQUENCE {triggerId TriggerId,
                                 executionParameterList ExecutionParameterList}


ExecutionParameterList ::= CHOICE {executionParameter ExecutionParameter,
                                   sequentialExecutionList SEQUENCE OF
                                   ExecutionParameter,
                                   parallelExecutionList SET OF
                                   ExecutionParameter}

ExecutionParameter  ::= SEQUENCE  {scriptId ScriptId,
                                   scriptParameters SEQUENCE OF Attribute}

emptyParameterList   ExecutionParameterList  ::= sequentialExecutionList:{ }

ErrorCode ::= SET OF INTEGER {noError(0),
                              noScriptError(1),
                              scriptRejectedError(2),
                              invalidParameterTypeError(3),
                              invalidParameterValueError(4),
                               scriptSyntaxError(5),
                              scriptExecutionFailedError(6),
                              invalidParmeterNumber(7),
                              unauthorizedAccessError(8)}
ScriptLanguageName ::=  OBJECT IDENTIFIER
ScriptContent  ::=  GeneralString
ModificationList ::= SET OF SEQUENCE{modifyOperator  [2] IMPLICIT
                                     ModifyOperator DEFAULT replace,
                                     attributeId AttributeId,
                                     attributeValue ANY DEFINED BY
                                     attributeId  OPTIONAL
                                      -- absent pour   setToDefault
                                    }


 END
```

Annexe B

Modèle relationnel général

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Le modèle GRM du séquenceur de commandes est décrit ci-après.

```
--
-- Les classes de relations suivantes prennent en charge le modèle du séquenceur
-- de commandes
--
commandSequencer-launchPadRelationshipClassBehaviour
BEHAVIOUR DEFINED AS

      !
            La classe de relation traite du rapport entre un séquenceur de commandes et
            ses pas de lancement utilisés pour amorcer l'exécution des scripts au moyen
            de tâches élémentaires nécessitant les services de support fournis par le
            séquenceur de commandes.

      !;


commandSequencer-LaunchPad-bindingBehaviour

BEHAVIOUR DEFINED AS

      !
            Cette notification a lieu lorsqu'il y a affectation d'un pas de lancement
            dans la relation séquenceur de commandes/pas de lancement.

      !;


commandSequencer-LaunchPad-unbindingBehaviour

BEHAVIOUR DEFINED AS

      !
            Cette notification a lieu lorsqu'un pas de lancement est retiré de la
            relation séquenceur de commandes/pas de lancement. Un pas de lancement ne
            peut être retiré de la relation que si son état administratif est
            verrouillé.

      !;


commandSequencer-launchPad-RelationshipClass
RELATIONSHIP CLASS

      BEHAVIOUR commandSequencer-launchPadRelationshipClassBehaviour,

                commandSequencer-LaunchPad-bindingBehaviour,

                commandSequencer-LaunchPad-unbindingBehaviour;

      SUPPORTS

            ESTABLISH,

            QUERY,

            TERMINATE,

            NOTIFY commandSequencer-LaunchPad-binding,

            NOTIFY commandSequencer-LaunchPad-unbinding;

      ROLE commandSequencerRole

            COMPATIBLE-WITH     commandSequencer

            PERMITTED-ROLE-CARDINALITY CONSTRAINT CASN1.RangeFromOneToOne

            REQUIRED-ROLE-CARDINALITY-CONSTRAINT CASN1.RangeFromOneToOne
```

```
            PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

            CASN1.RangeFromOneToOne

            REGISTERED AS  { CSModule.cmdSeqRelationshipRoles 1 }


      ROLE    launchPadRole

            COMPATIBLE-WITH launchPad

            PERMITTED-ROLE-CARDINALITY-CONSTRAINT   CSModule.RangeFromZeroToMax

            REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromZeroToMax

            BIND-SUPPORT

            UNBIND-SUPPORT

            PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne

      REGISTERED AS { CSModule.cmdSeqRelationshipRoles 2 };

REGISTERED AS { CSModule.cmdSeqRelationshipClasses 1 };


commandSequencer-LaunchPad-RelationshipMapping-Behaviour

BEHAVIOUR DEFINED AS

      !

            Cette mise en correspondance de relation décrit la manière dont la classe
            relation entre séquenceur de commandes et pas de lancement peut être
            représentée en utilisant le confinement. Dans cette mise en correspondance
            de relation, le séquenceur de commandes est l'objet d'ordre supérieur aux
            fins de dénomination et il contient les pas de lancement. La participation à
            cette relation implique que le pas de lancement et son générateur peuvent
            utiliser les services fournis par la ressource représentée par le séquenceur
            de commandes.

      !;


commandSequencer-launchPad-RelationshipMapping

RELATIONSHIP MAPPING

      RELATIONSHIP CLASS
      commandSequencer-launchPad-RelationshipClass;

      BEHAVIOUR commandSequencer-launchPad-RelationshipMapping-Behaviour;


      ROLE commandSequencerRole

            RELATED-CLASSES commandSequencer
            REPRESENTED BY NAMING

                  launchPad-commandSequencer-NameBinding
                  USING SUPERIOR,


      ROLE launchPadRole

            RELATED-CLASSES launchPad
            REPRESENTED BY NAMING

                  launchPad-commandSequencer-NameBinding
                  USING SUBORDINATE;


      OPERATIONS  MAPPING

            ESTABLISH     MAPS-TO-OPERATION

                          CREATE commandSequencer OF commandSequencerRole

            TERMINATE     MAPS-TO-OPERATION

                          DELETE commandSequencer OF commandSequencerRole
```

```
            NOTIFY commandSequencer-launchPad-binding

                        MAPS-TO-OPERATION

                        NOTIFICATION "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":
                        objectCreationNotification

                              OF commandSequencerRole

            NOTIFY commandSequencer-launchPad-unbinding

                        MAPS-TO-OPERATION

                        NOTIFICATION "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":
                        objectDeletionNotification

                              OF commandSequencerRole

            BIND   MAPS-TO-OPERATION

                        CREATE launchPad OF launchPadRole

            UNBIND       MAPS-TO-OPERATION

                        DELETE launchPad

                              OF launchPadRole

            QUERY  MAPS-TO-OPERATION

                        GET OF commandSequencerRole

                        GET OF launchPadRole;

REGISTERED AS {CSModule.cmdSeqRelationshipMappings 1}


scriptReferenceRelationshipClassBehaviour

BEHAVIOUR DEFINED AS

      !

            Cette classe de relation décrit la relation existant entre des scripts et un
            objet qui fait référence à ces scripts afin d'identifier la (les) tâche(s) à
            effectuer.

      !;


scriptReferencerRelationshipClass

RELATIONSHIP CLASS

      BEHAVIOUR scriptReferencerRelationshipClassBehaviour;
      SUPPORTS

            ESTABLISH,
            TERMINATE,
            QUERY;


      ROLE scriptUserRole

            COMPATIBLE-WITH scriptReferencer

            PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne

            REQUIRED-ROLE-CARDINALITY CONSTRAINT CSModule.RangeFromOneToOne

            PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

                  CSModule.RangeFromZeroToMax

      REGISTERED AS {CSModule.cmdSeqRelationshipRoles 3}


      ROLE scriptRole

            COMPATIBLE-WITH launchScript

            PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne

            REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne
```

```
            PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

                  CSModule.RangeFromZeroToMax

      REGISTERED AS {CSModule.cmdSeqRelationshipRoles 4}


REGISTERED AS {CSModule.cmdSeqRelationshipClasses 2}


launchPad-launchScriptRelationshipMappingBehaviour BEHAVIOUR

DEFINED AS

!
      Cette mise en correspondance de relation décrit la relation entre le pas de
      lancement et le script de lancement. Le pas de lancement amorce l'exécution du
      script de lancement et y fait référence aux fins d'exécution.

!;


launchPad-LaunchScriptMapping
RELATIONSHIP MAPPING

       RELATIONSHIP CLASS scriptReferenceRelationshipClass;

       BEHAVIOUR launchPad-launchScriptMappingBehaviour;

       ROLE scriptUserRole

           RELATED CLASSES launchPad

           REPRESENTED BY ATTRIBUTE scriptList

       ROLE   scriptRole

           RELATED CLASSES launchScript;

       OPERATIONS MAPPING

           ESTABLISH MAPS-TO-OPERATION

               CREATE launchPad OF scriptUserRole

               -- en utilisant SET-BY-CREATE de scriptList --

               REPLACE scriptList of scriptUserRole

               -- ce qui ajoute effectivement un scriptId à la liste scriptList --

               ADD scriptList of scriptUserRole,

           TERMINATE MAPS-TO-OPERATION

               DELETE launchPad OF scriptUserRole

               REPLACE scriptList OF scriptUserRole

               -- ce qui retire effectivement un scriptId de la liste scriptList --

               REMOVE scriptList OF scriptUserRole,

           QUERY MAPS-TO-OPERATION

               GET scriptList OF scriptUserRole;

REGISTERED AS {CSModule.cmdSeqRelationshipMappings 2};


thread-launchScriptRelationshipMappingBehaviour BEHAVIOUR

DEFINED AS

!
      Cette classe de relation décrit le rapport entre tâche élémentaire et script de
      lancement. Le script de lancement est référencé par la tâche élémentaire au moyen
      de l'attribut scriptId contenu dans la liste scriptList.

!;


thread-launchScriptMapping
RELATIONSHIP MAPPING
```

```
        RELATIONSHIP CLASS scriptReferenceRelationshipClass;
        BEHAVIOUR thread-launchScriptRelationshipMappingBehaviour;
        ROLE scriptUserRole
            RELATED CLASSES thread
            REPRESENTED BY ATTRIBUTE scriptId
            QUALIFIED BY scriptList
        ROLE   scriptRole
            RELATED CLASSES launchScript;
        OPERATIONS MAPPING
            ESTABLISH MAPS-TO-OPERATION
                CREATE OF scriptUserRole,
            TERMINATE MAPS-TO-OPERATION
                DELETE OF scriptUserRole,
            QUERY MAPS-TO-OPERATION
                GET scriptId OF scriptUserRole;
REGISTERED AS {CSModule.cmdSeqRelationshipMappings 3};


spawner-progeny-RelationshipClass
RELATIONSHIP CLASS
        BEHAVIOUR spawner-progenyRelationshipClassBehaviour BEHAVIOUR
       DEFINED AS
              !
              Lorsqu'une instance de cette classe donne lieu à la création de nouveaux
              objets, elle sert d'objet IVMO pendant la création en fournissant des
              valeurs d'attributs pour le nouvel objet sur la base de son propre attribut
              defaultExecutionParameterList et d'éventuels paramètres qui ont été fournis
              dans le cadre de l'action ou des instances locales de cette classe qui sont
              capables de donner lieu à la création de nouvelles instances d'objets. Les
              instances nouvellement créées seront contenues par cette instance et leurs
              noms seront générés automatiquement. L'état d'utilisation de l'objet sera
              "en cours d'utilisation" jusqu'à ce que tous les objets créés soient
              terminés. Si l'état administratif de cet objet est "verrouillé", ces
              nouveaux objets ne pourront pas être créés. Si, du fait d'une limitation des
              ressources locales, cet objet est incapable de prendre en charge un plus
              grand nombre d'objets contenus, son état d'utilisation sera "occupé".


              Une instance de cette classe peut donner lieu à la création de nouvelles
              instances d'objets à partir d'un simple scriptId, en utilisant un ensemble
              de scriptIds dans n'importe quel ordre ou à partir d'une séquence de
              scriptIds dans l'ordre spécifié dans la liste, c'est-à-dire qu'après
              création de la première instance, la seconde ne peut être créée qu'une fois
              terminée la première instance, etc. La valeur de l'attribut scriptId de
              l'objet créé tire sa valeur de l'élément correspondant de la liste
              scriptList de l'objet.
            ;
        SUPPORTS
            ESTABLISH
            TERMINATE;


        ROLE spawnerRole COMPATIBLE-WITH basicSpawnerClass
            PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeOneToOne
            REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeOneToOne
```

```
            PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

                CSModule.RangeOneToOne

            REGISTERED AS {CSModule.cmdSeqRoles 5}

        ROLE progenyRole COMPATIBLE-WITH thread

            PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeZeroToMax

            REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeZeroToMax

            PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT
        CSModule.RangeOneToOne

            REGISTERED AS {CSModule.cmdSeqRoles 6}

REGISTERED AS {CSModule.cmdSeqRelationshipClasses 3};
```

Annexe C

Définitions des informations de gestion pour le comptage de discrimination d'événements

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

L'objet de classe EDC (compteur de discrimination d'événements) compte le nombre d'événements d'entrée. Avant le comptage, l'EDC teste les valeurs d'attributs liés à l'événement ou à un objet générant l'événement et effectue une discrimination de l'événement.

C.1 Classe d'objets gérés

```
eventDiscriminationCounter MANAGED OBJECT CLASS

DERIVED FROM   "DMI":discriminator;

CHARACTERIZED BY

edcPackage PACKAGE

        BEHAVIOUR edcBehaviour BEHAVIOUR

       DEFINED AS

       "Si le résultat de la discrimination d'un rapport potentiel d'événement est évalué
       à TRUE (vrai) et si le compteur de discrimination d'événements est à l'état
       déverrouillé et activé et ne présente pas l'état de disponibilité hors service,
       dans ce cas, la valeur de l'attribut compteur est incrémentée.";;

              ATTRIBUTES

                    "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":counter GET,

                    maxCounterSize GET;

              NOTIFICATIONS

                    "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":processingErrorAlarm;;;

CONDITIONAL PACKAGES

       counterAlarmPackage PRESENT IF "un compteur est d'une taille finie et une
       notification est déclenchée par un seuil d'alarme de capacité.";

REGISTERED AS {joint-iso-itu-t ms(9) ms(9) function(2) part21(21)

managedObjectClass(3) xx11(11)};
```

C.2 Bloc

```
counterAlarmPackage PACKAGE

        BEHAVIOUR

        counterAlarmBehaviour BEHAVIOUR

        DEFINED AS "Lorsque la valeur de compteur atteint le seuil d'alarme de capacité
               (en pourcentage de la taille maximale du compteur) le compteur EDC génère
               un événement indiquant qu'un seuil de capacité a été atteint ou dépassé.
               En signalant l'événement de seuil de capacité, le rapport d'alarme défini
               dans la Rec. X.733 du CCITT | ISO/CEI 10164-4 est utilisé. Seuls, les
               paramètres suivants du rapport d'alarme doivent être utilisés et tous les
               paramètres sont obligatoires lorsqu'ils sont utilisés pour signaler des
               alarmes de seuil de capacité du compteur.

                Classe d'objets gérés - Ce paramètre doit identifier la classe de

                compteur.

                Instance d'objets gérés - Ce paramètre doit identifier l'instance du

                compteur qui a généré l'événement.
```

Type d'alarme - Ce paramètre doit indiquer qu'une alarme d'erreur de traitement a eu lieu.

Heure de l'événement - Ce paramètre contient l'heure à laquelle l'événement de seuil de capacité a eu lieu.

Gravité perçue - Ce paramètre indiquera la gravité attribuée à l'événement de seuil de capacité. Lorsque la condition compteur plein à 100% est atteinte, une valeur de gravité critique doit être attribuée à cet événement.

Attributs surveillés - Ce paramètre contiendra l'attribut taille maximale du compteur pour l'EDC.

Cause probable - Ce paramètre doit contenir la valeur de congestion.

Information de seuil - Ce paramètre doit contenir la valeur seuil de capacité (en pourcentage de la capacité totale) qui a été atteinte ou dépassée lors de la génération de cet événement.";;

ATTRIBUTES

"Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":capacityAlarmThreshold GET-REPLACE ADD-REMOVE;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx15(15)};

C.3 Attribut

maxCounterSize ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSModule.MaxCounterSize;

MATCHES FOR EQUALITY, ORDERING;

BEHAVIOUR

maxSizeOrderingBehaviour BEHAVIOUR

DEFINED AS "Cet attribut représente la valeur la plus élevée du compteur. L'ordre est le même que pour les entiers positifs qui augmentent séquentiellement sauf que la valeur 0 est la plus grande et représente une taille infinie.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx12(12)};

Annexe D

Classe d'objets support de gestion cmisScript

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Il s'agit d'une classe d'objets gérés script, le script CMIS (service commun d'information de gestion), qui peut être définie pour traiter l'invocation des opérations CMIS.

D.1 Attributs

D.1.1 Attributs tirés de la définition des informations de gestion

La présente Spécification fait référence aux attributs de gestion suivants dont la syntaxe abstraite est spécifiée dans la Rec. X.721 du CCITT | ISO/CEI 10165-2:

a) attributeIdentifierList;

b) objectClass;

c) attributeList.

La présente Spécification fait référence aux attributs de gestion suivants dont la syntaxe abstraite est spécifiée dans la Rec. UIT-T X.711 | ISO/CEI 9596-1:

a) synchronization;

b) scope;

c) filter;

d) baseManagedObjectId;

e) modificationList.

D.2 Définitions

D.2.1 **cmisScript:** objet support de gestion, qui dirige l'invocation d'une opération CMIS unique.

Cinq types de scripts CMIS sont spécifiés:

D.2.2 **getCmisScript:** script CMIS qui représente une opération GET.

D.2.3 **setCmisScript:** script CMIS qui représente une opération SET.

D.2.4 **actionCmisScript:** script CMIS qui représente une opération ACTION.

D.2.5 **createCmisScript:** script CMIS qui représente une opération CREATE.

D.2.6 **deleteCmisScript:** script CMIS qui représente une opération DELETE.

Les scripts CMIS sont des scripts à une seule instruction qui donnent des détails opérationnels aux tâches élémentaires. Par exemple, les langages de description peuvent faire référence à des enregistrements de commande appropriés afin de faire une demande CMIS à un agent.

D.3 getCmisScript

D.3.1 Caractéristiques de getCmisScript

Les attributs suivants sont définis dans le script getCmisScript:

– baseManagedObjectId;

– synchronization;

– scope;

– filter;

– attributeIdentifierList.

D.3.2 Blocs de getCmisScript

getCmisScript contient le bloc obligatoire suivant:

- getCmisScriptPackage.

D.4 setCmisScript

D.4.1 Caractéristiques de setCmisScript

setCmisScript dispose des définitions d'attribut suivantes:

- baseManagedObjectId;
- synchronization;
- scope;
- filter;
- modificationList.

D.4.2 Blocs de setCmisScript

setCmisScript contient le bloc obligatoire suivant:

- setCmisScriptPackage.

D.5 actionCmisScript

D.5.1 Caractéristiques de actionCmisScript

actionCmisScript dispose des définitions d'attribut suivantes:

- baseManagedObjectId;
- synchronization;
- scope;
- filter.

D.5.2 Blocs de actionCmisScript

La classe d'enregistrement de commandes d'action contient le bloc obligatoire suivant:

- actionCmisScriptPackage.

D.6 createCmisScript

D.6.1 Caractéristiques de createCmisScript

L'enregistrement de commande d'action dispose des définitions d'attribut suivantes:

- objectClass;
- attributeList.

D.6.2 Blocs de createCmisScript

La classe d'enregistrement de commandes de création contient le bloc obligatoire suivant:

- createCmisScriptPackage.

Elle contient les blocs conditionnels suivants:

- managedObjectInstancePackage;
- superiorObjectInstancePackage;
- referenceObjectInstancePackage.

D.7 deleteCmisScript

D.7.1 Caractéristiques de deleteCmisScript

deleteCmisScript dispose des définitions d'attribut suivantes:

- baseManagedObjectId;
- synchronization;
- scope;
- filter.

D.7.2 Blocs de deleteCmisScript

deleteCmisScript contient le bloc obligatoire suivant:

- deleteCmisScriptPackage.

D.8 Services

D.8.1 Définition du service de rapport Get

Le présent paragraphe spécifie le service de rapport Get et le mappe sur les services M-EVENT-REPORT du CMIS.

D.8.2 Définition du service de rapport Set

Le présent paragraphe spécifie le service de rapport Set et le mappe sur les services M-EVENT-REPORT du CMIS.

D.8.3 Définition du service de rapport d'action

Le présent paragraphe spécifie le service de rapport action et le mappe sur les services M-EVENT-REPORT du CMIS.

D.8.4 Définition du service de rapport de création

Ce service permet à un utilisateur MIS, jouant le rôle d'agent, de rapporter la création d'un objet géré. Il est défini comme un service à la fois confirmé et non confirmé et est mappé sur les services M-EVENT-REPORT du CMIS. Ce service est défini dans la Rec. X.730 du CCITT | ISO/CEI 10164-1.

D.8.5 Définition du service de rapport de suppression

Ce service permet à un utilisateur MIS, jouant le rôle d'agent, de rapporter la suppression d'un objet géré. Il est défini comme un service à la fois confirmé et non confirmé et est mappé sur les services M-EVENT-REPORT du CMIS. Ce service est défini dans la Rec. X.730 du CCITT | ISO/CEI 10164-1.

D.9 Modèle GDMO

D.9.1 Définitions des objets gérés

```
cmisScript MANAGED OBJECT CLASS
       DERIVED FROM launchScript;
CHARACTERIZED BY cmisScriptPackage PACKAGE
        BEHAVIOUR
              cmisScriptBehaviour BEHAVIOUR
       DEFINED AS
       !
              Une instance de cette classe d'objets gérés modélise les informations
              nécessaires à l'exécution d'une opération CMIS unique.
       !;;;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx12(12)};
getCmisScript MANAGED OBJECT CLASS
       DERIVED FROM cmisScript;
CHARACTERIZED BY getCmisScriptPackage PACKAGE
        BEHAVIOUR
```

```
       getCmisScriptBehaviour   BEHAVIOUR
       DEFINED AS
       !
              Une instance de cette classe d'objets gérés modélise les informations
              nécessaires à l'exécution d'une opération GET CMIS unique.
       !;;
       ATTRIBUTES
              baseManagedObjectId GET,
              synchronization GET-REPLACE,
              scope GET-REPLACE,
              filter GET-REPLACE,
              "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992":attributeIdentifierList
               GET-REPLACE ADD-REMOVE;;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx13(13)};


setCmisScript MANAGED OBJECT CLASS
       DERIVED FROM cmisScript;
CHARACTERIZED BY setCmisScriptPackage PACKAGE
        BEHAVIOUR
       setCmisScriptBehaviour BEHAVIOUR
       DEFINED AS
       !
              Une instance de cette classe d'objets gérés modélise les informations
              nécessaires à l'exécution d'une opération SET CMIS unique.
       !;;
       ATTRIBUTES
              baseManagedObjectId GET,
              synchronization      GET-REPLACE,
              scope                GET-REPLACE,
              filter               GET-REPLACE,
              modificationList     GET-REPLACE ADD-REMOVE;;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx14(14)};


actionCmisScript MANAGED OBJECT CLASS
       DERIVED FROM cmisScript;
       CHARACTERIZED BY
actionCmisScriptPackage PACKAGE BEHAVIOUR
              actionCmisScriptBehaviour BEHAVIOUR
              DEFINED AS
              !
              Une instance de cette classe d'objets gérés modélise les informations
              nécessaires à l'exécution d'une opération ACTION CMIS unique.
              !;;
```

```
        ATTRIBUTES

                baseManagedObjectId GET,

                synchronization     GET-REPLACE,

                scope               GET-REPLACE,

                filter              GET-REPLACE;;;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx15(15)};


createCmisScript  MANAGED OBJECT CLASS
DERIVED FROM cmisScript;
CHARACTERIZED BY

       createCmisScriptPackage PACKAGE

       BEHAVIOUR

              createCmisScriptBehaviour BEHAVIOUR

              DEFINED AS

              !
                     Une instance de cette classe d'objets gérés modélise les informations
                     nécessaires à l'exécution d'une opération CREATE CMIS unique.

              !;;

       ATTRIBUTES

              "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": objectClass   GET,

              "Rec. X.721 du CCITT | ISO/CEI 10165-2:1992": attributeList GET-REPLACE
              ADD-REMOVE;;;

       CONDITIONAL PACKAGES

              managedObjectInstancePackage

                     PRESENT IF "le superiorObjectInstancePackage est absent.",

              superiorObjectInstancePackage

                     PRESENT IF "le managedObjectInstancePackage est absent.",

              referenceObjectInstancePackage

                     PRESENT IF "le gestionnaire a la valeur spécifiée.";

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx16(16)};


deleteCmisScript  MANAGED OBJECT CLASS
DERIVED FROM  cmisScript;
CHARACTERIZED BY

       deleteCmisScriptPackage PACKAGE

       BEHAVIOUR

       deleteCmisScriptBehaviour BEHAVIOUR
       DEFINED AS

       !
              Une instance de cette classe d'objets gérés modélise les informations
              nécessaires à l'exécution d'une opération DELETE CMIS unique.

       !;;

       ATTRIBUTES

              baseManagedObjectId GET,
              synchronization     GET-REPLACE,
              scope               GET-REPLACE,
              filter              GET-REPLACE;;;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx17(17)};
```

D.9.2 Définitions des blocs

```
 managedObjectInstancePackage   PACKAGE
  ATTRIBUTES
        managedObjectInstance   GET-REPLACE;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx16(16)};


superiorObjectInstancePackage   PACKAGE
        ATTRIBUTES
        superiorObjectInstance   GET-REPLACE;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx17(17)};


       referenceObjectInstancePackage  PACKAGE
        ATTRIBUTES
        referenceObjectInstance  GET-REPLACE;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx18(18)};
```

D.9.3 Définitions des attributs

```
baseManagedObjectId   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX CSModule.BaseManagedObjectId;
        MATCHES FOR EQUALITY;
        BEHAVIOUR baseManagedObjectIdBehaviour
        BEHAVIOUR
        DEFINED AS
       !
           Il s'agit de l'identificateur des informations sur l'opération CMIS
             à exécuter.
        !;;
        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx13(13)
};


scope ATTRIBUTE
        WITH ATTRIBUTE SYNTAX CSModule.Scope;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
        scopeBehaviour     BEHAVIOUR
        DEFINED AS
        !
              Il s'agit de la première phase de sélection de l' (des) objet(s) géré(s)
              vers lequel (lesquels) il convient de diriger les opérations de script CMIS.
              Cet attribut indique l' (les) objet(s) auquel (auxquels) il convient
              d'appliquer un filtre.
        !;;
         REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7)
xx14(14)};
```

```
filter ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.CMISFilter;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        filterBehaviour    BEHAVIOUR

        DEFINED AS

        !

              Il s'agit de la seconde phase de sélection d'objet(s) géré(s) vers lequel
              (lesquels) il convient de diriger les opérations de script CMIS. Il est
              appliqué un ensemble de tests à chacun des objets gérés sélectionnés par la
              détection pour en extraire un sous-ensemble.

        !;;

        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx15(15)
};


synchronization ATTRIBUTE

       WITH ATTRIBUTE SYNTAX CSModule.CMISSync;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        synchronizationBehaviour BEHAVIOUR

        DEFINED AS

        !

              Cet attribut indique la manière dont les opérations doivent être
              synchronisées entre des instances (occurrences) d'objets gérés lorsque
              plusieurs objets gérés ont été sélectionnés par les mécanismes de détection
              et de filtrage.

        !;;

        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx16(16)
};


modificationList ATTRIBUTE

       WITH ATTRIBUTE SYNTAX CSModule.ModificationList;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        modificationListBehaviour BEHAVIOUR

        DEFINED AS

        !

              Cet attribut représente la liste des attributs à modifier par le script
              CMISSet et contient les valeurs auxquelles ces attributs doivent être
              fixés.”;;


        !;;

        REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx17(17)
};


superiorObjectInstance ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.ObjectInstance;

        MATCHES FOR EQUALITY;

        BEHAVIOUR
```

```
        superiorObjectInstanceBehaviour BEHAVIOUR

        DEFINED AS

        !

            Cet attribut identifie l'instance d'objets gérés existante qui est

            fournie, l'attribut managedObjectInstance ne doit pas être fourni.

        !;;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx18(18)};


referenceObjectInstance ATTRIBUTE

        WITH ATTRIBUTE SYNTAX CSModule.ObjectInstance;

        MATCHES FOR EQUALITY;

        BEHAVIOUR

        referenceObjectInstanceBehaviour BEHAVIOUR

        DEFINED AS

        !

            Le nom de l'instance d'objets gérés de la même classe que l'objet géré à créer.
            Les valeurs d'attribut associées à l'instance d'objet de référence deviennent
            les valeurs par défaut pour celles qui ne sont pas spécifiées par l'attribut
            liste d'attributs.



        !;;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx19(19)};
```

Annexe E

Classe d'objets gérés CMIP_CS

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe fournit un exemple de la manière dont les sous-classes supplémentaires peuvent être définies pour contenir des attributs tels que le titre AE.

E.1 cmipCS

E.1.1 Aperçu général

– Sous-classe de l'automate protocolaire cmip et du séquenceur de commandes.

– Définit l'invocation du titre AE pour le séquenceur de commandes.

E.1.2 Caractéristiques de la classe d'objets gérés cmipCS

– aetitle.

E.1.3 Blocs de cmipCS

La classe d'objets gérés cmipCS contient le bloc obligatoire suivant:

– bloc aetitle.

E.1.4 Définitions GDMO

```
cmipCS MANAGED OBJECT CLASS
DERIVED FROM commandSequencer;
CHARACTERIZED BY aeTitlePackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx18(18)};


aeTitlePackage PACKAGE
ATTRIBUTES
           aetitle GET;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx19(19)};
```

E.1.5 Définitions des attributs

```
aetitle ATTRIBUTE
        WITH ATTRIBUTE SYNTAX CSModule.AE-title;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
        aeTitleBehaviour BEHAVIOUR
       DEFINED AS "Une instance de cette classe d'objets gérés définit l'appel du
       titre AE pour le séquenceur de commandes";;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx20(20)};
```

Annexe F

Langage de scriptage de gestion-systèmes (SMSL)

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe définit un langage de scriptage à chaîne générale, le langage SMSL, permettant d'écrire des procédures d'exécution dans un environnement de séquenceur de commandes.

Le langage SMSL dispose d'un certain nombre de fonctions nécessaires à un tel environnement. Le langage SMSL utilise ses fonctions aux dépens de la complétude des langages tels que C, csh, Perl ou awk tout en utilisant certaines des instructions et fonctions qui font la puissance et le succès de ces langages.

F.1 Mappage des directives GDMO sur le langage SMSL

Le SMSL dispose d'un automate virtuel capable d'interpréter les déclarations GDMO, telles que celles-ci sont décrites dans les Normes ISO/CEI 10165 et 10164; ces structures GDMO ont été mappées sur les types de données SMSL conformément aux modèles d'objet SMSL décrits en F.14.11. Les scripts écrits en langage SMSL sont interprétés et exécutés par l'automate virtuel SMSL. Dans l'environnement du séquenceur de commandes, l'automate virtuel SMSL joue le rôle de gestionnaire aux fins de la gestion-systèmes. L'automate virtuel SMSL doit être capable d'interpréter tous les types de données SMSL utilisés par les scripts SMSL, ce qui signifie, du point de vue du rédacteur du script, que tous les types utilisés dans le script sans les définitions de ces types sont implicitement connus par l'automate virtuel. L'automate virtuel SMSL doit être capable d'interpréter toutes les instances SMSL déjà instanciées, ce qui signifie que, du point de vue du rédacteur du script, toutes les références à des objets précédemment instanciés sont connues par l'automate virtuel.

F.2 Fonctions intégrées SMSL

Le langage SMSL comprend un certain nombre de fonctions intégrées pour la création et la manipulation d'objets, ainsi que des fonctions d'usage général telles que des fonctions mathématiques, logiques et d'entrée/sortie. Un résumé des fonctions intégrées du langage SMSL est fourni ci-après. Les fonctions sont décrites séparément.

F.2.1 Fonctions de contrôle de concomitance

Le langage SMSL comprend deux fonctions intégrées permettant de réaliser le contrôle de concomitance: verrouillé() et déverrouillé(). Ces fonctions sont en général utilisées pour permettre un accès linéaire aux structures et aux ressources de données partagées.

Tous les processus SMSL qui tentent de linéariser des accès à une ressource doivent coopérer en demandant des verrouillages d'une dénomination de verrou donnée. L'accès à des ressources partagées est refusé à toutes les fonctions, y compris des fonctions sur les ensembles set()-modifier et get()-obtenir, lorsque le verrou est absent. Il incombe à chaque processus SMSL de n'accéder à une ressource que lorsqu'il dispose du verrou exigé.

F.3 Fonctions sur les ensembles pour des listes SMSL

Le langage SMSL comprend les fonctions suivantes pour exécuter des opérations d'ensembles sur des listes SMSL:

- différence() pour extraire la liste des éléments différents entre listes;
- intersection() pour extraire une liste d'éléments communs entre listes;
- tri() pour extraire une liste d'éléments en ordre croissant ou décroissant;
- sous-ensemble() pour s'assurer qu'une liste est contenue dans une autre;
- union() pour extraire une liste qui est une combinaison de listes;
- unique() pour extraire une liste d'éléments qui apparaissent dans une seule liste.

Ces fonctions traitent les listes SMSL comme des ensembles d'éléments. Chaque membre d'une liste est une chaîne de texte qui se termine par un caractère espace.

L'ensemble NULL [""] est l'équivalent de l'ensemble nul ou vide (Ø) dans la théorie des ensembles. L'ensemble NULL est traité par les fonctions sur les ensembles SMSL comme un ensemble à part entière qui ne contient pas d'élément. La chaîne NULL [ ] est un élément de la liste SMSL sans caractère. Les fonctions sur les ensembles SMSL permettent l'utilisation de chaînes NULL dans les listes.

Le concept SMSL d'un ensemble n'est pas la liste unique d'éléments croissants ou décroissants tel qu'il est défini dans la théorie des ensembles. Dans de nombreux cas, les listes SMSL contiennent des éléments dupliqués sans ordre particulier. Une liste SMSL peut être transformée en un ensemble ordonné en utilisant la fonction unique() pour retirer les éléments en double et la fonction tri() pour disposer les éléments en ordre croissant ou décroissant.

F.4 Fonctions mathématiques du langage SMSL

Le langage SMSL prend en charge un sous-ensemble de fonctions mathématiques de base. Ces fonctions sont toutes semblables aux fonctions mathématiques C.

Les fonctions mathématiques SMSL comprennent un certain niveau de vérification d'erreurs à l'exécution en termes de plage et de domaine. Ces deux conditions donnent lieu à un message d'erreur à l'exécution qui établit la valeur appropriée d'une variable errno du langage SMSL.

En outre, toute valeur non numérique produite par l'impression du résultat de l'invocation de la fonction tel que Nan ou -Inf est convertie à 0.0 pour éviter que la valeur retournée ne soit interprétée par le langage SMSL comme une chaîne de caractères non numériques. La fonction SMSL renvoie également un message d'erreur à l'exécution lorsqu'elle effectue la conversion.

F.5 Synchronisation de processus SMSL

Le langage SMSL assure une synchronisation des processus au sein des processus SMSL d'un même séquenceur de commandes par le biais de primitives de variables conditionnelles utilisées avec des verrous SMSL. Ces primitives sont semblables à la construction fournie pour la programmation multitransactionnelle en langage de programmation C sur des systèmes d'exploitation non transactionnels.

F.6 Canaux globaux partagés SMSL

Le langage SMSL prend en charge l'utilisation de canaux globaux partagés pour la communication entre un procédé et un autre procédé ou fichier.

Le SMSL permet à un procédé SMSL d'ouvrir un canal vers un procédé externe selon un mode explicitement partagé qui permet à tout nombre d'autres procédés SMSL d'envoyer des données au canal et d'en recevoir.

La capacité de partager des canaux exige que le langage SMSL dispose également d'un mécanisme de techniques de programmation concomitante. Le langage SMSL fournit ces techniques de programmation sous la forme de primitives de synchronisation externes.

Il est préférable d'utiliser des primitives plutôt que de construire la concurrence dans les fonctions d'ouverture, de lecture, d'écriture et de fermeture du canal. Par exemple, le fait de faire en sorte que chaque processus SMSL verrouille un canal partagé, évite de manière explicite la concomitance de lecture par un procédé pendant l'écriture par un autre. En outre, la séparation entre les primitives de synchronisation et les canaux permet de les utiliser pour synchroniser l'accès à d'éventuelles ressources partagées telles que la table des symboles internes de l'agent ou un fichier externe.

Les fonctions read()-lire, readln()-lireln et write()-écrire pour des canaux partagés échoueront immédiatement (sans blocage) si un autre processus SMSL est déjà bloqué sur le canal.

F.6.1 L’effet des mécanismes de canaux globaux partagés SMSL

Les fonctions SMSL assurent une sérialisation de toutes les opérations sur un canal donné et, dans ce cas, tous les appels à des fonctions SMSL semblent être atomiques. Le programmeur utilisant le langage SMSL peut être sûr que les lectures et écritures de canaux de fichiers dans différents processus ont lieu de manière atomique. Les verrous fournis dans le langage SMSL évitent l'entrelaçage imprévisible de séquences d'invocation de lecture et d'écriture SMSL vers le canal.

La seule exception à la sérialisation sur des canaux créés au moyen de la fonction popen() est qu'elle autorise une concomitance d'opérations de lecture et d'écriture. Il peut y avoir lecture alors qu'une écriture est en cours sur le canal et il peut y avoir écriture alors qu'une lecture est bloquée ou en cours sur le canal, ainsi des processus SMSL de lecture et d'écriture peuvent être bloqués sur un canal partagé.

Les canaux de fichiers ouverts au moyen de la fonction fopen() ne peuvent jamais donner lieu à un blocage de fonction de lecture ou d'écriture SMSL. Pour mettre la sérialisation en application, ni le second procédé de lecture ni le second procédé d'écriture ne peuvent être bloqués; par conséquent, la seconde fonction SMSL read(), readln() et write() sur un canal de fichier échouera.

F.7 Types de données et d'objets SMSL

Le langage SMSL dispose de quatre types de données de base: entier, décimal, chaîne et liste. Les valeurs de ces quatre types simples peuvent être manipulées comme s'il s'agissait de chaînes de caractères. Les types de données complexes peuvent être constitués au moyen de structs et de constructions matricielles. Le langage SMSL prend en charge tous les types de la syntaxe ASN.1.

F.7.1 Conversion entre type GDMO ASN.1, d'une part, et type orienté-script SMSL, d'autre part

Le tableau suivant donne le rapport entre un type GDMO ASN.1, d'une part, et le type SMSL correspondant, d'autre part.

Tableau F.1 – Conversion des valeurs entre type ASN.1 et type orienté-script

Groupe de types de valeur	Type GDMO ASN.1	Type orienté-script
Groupe entier	ENTIER, BOOLÉEN	entier
Groupe décimal	RÉEL	à virgule flottante
Chaîne	Chaîne générale	chaîne
Groupe d'ensembles	SÉQUENCE, ENSEMBLE	type d'objet
Expression matricielle	SÉQUENCE DE, ENSEMBLE DE	liste
Groupe d'affectation de noms	ObjectInstance	type d'objet

Les variables et les valeurs sont interprétées soit comme des chaînes, soit comme des nombres, selon le contexte.

Une grandeur scalaire (entière ou décimale) est interprétée comme vraie au sens booléen du terme s'il ne s'agit pas de la chaîne NULL (vide) ou 0. Les booléens renvoyés par les opérateurs sont 1 pour vrai et 0 ou "" (la chaîne nulle) pour faux.

F.7.2 Constantes numériques

Bien que la représentation interne d'une constante entière ou réelle soit une chaîne, il n'est pas nécessaire que ces constantes apparaissent entre guillemets dans les scripts SMSL. Quelques exemples de constantes entières et réelles SMSL sont fournis ci-après:

x = 3; pi = 3,14159;

Tableau F.2 – Exemples de types de données SMSL

Type de données	Exemple de représentation	Représentation SMSL
entier	3	"3"
décimal	4,5	"4.5"
chaîne	"abc"	"abc"
liste	[1,3,5]	"1\n3\n5"
NOTE – "\n" est le caractère d'interligne.		

F.7.3 Types de données complexes

Un struct est un ensemble de types de données qui peuvent être regroupés afin de représenter des structures ASN.1 complexes qui permettent de construire des ensembles et des séquences (SETs et SEQUENCEs ASN.1) en utilisant struct. Il est possible d'accéder à des éléments struct particuliers en utilisant l'opérateur "." .

Des expressions matricielles peuvent être représentées en utilisant le type liste dans le langage SMSL. L'affectation de noms est mise en correspondance avec la valeur de l'objet nom.

F.8 Variables SMSL

Des variables de tous types peuvent être utilisées comme lvalues – ce qui signifie qu'elles sont attribuables. Etant donné que tous les types de données sont traités en interne comme des chaînes, ils partagent tous un espace nom commun. Par conséquent, il est possible d'utiliser le même nom pour une variable scalaire, une variable chaîne et une variable liste.

Le cas décrit ici est particulièrement significatif. "FOO", "Foo" et "foo" sont tous des noms différents. Les noms doivent commencer par une lettre ou un caractère de soulignement mais peuvent contenir des chiffres et des caractères de soulignement ("_").

Certains identificateurs ont des significations prédéfinies. Des mots clés réservés – tels que if et foreach – ne peuvent pas être utilisés comme identificateurs. Les mots clés sont connus lorsqu'ils sont tous en lettres minuscules ou majuscules. En outre, il n'est pas possible d'utiliser des constantes prédéfinies en tant qu'identificateurs.

F.8.1 Initialisation par défaut des variables SMSL

Le langage SMSL n'utilise pas le concept de "déclarations" pour des variables. La première apparition d'un identificateur est utilisée pour ajouter cet identificateur à la liste des variables globales d'un script SMSL donné. Toutes les variables sont initialisées avec une valeur chaîne nulle à chaque fois qu'un script SMSL est exécuté. Cette valeur ne change qu'une fois la valeur de la variable définie par une quelconque opération explicite telle que attribution (assignment).

Cette initialisation par défaut sur la chaîne vide permet de traiter une variable comme une liste/chaîne initialement vide ou comme une variable numérique de valeur 0 (étant donné que les opérateurs arithmétiques considèrent la chaîne NULL (vide) comme équivalente à 0). Cependant, le fait de dépendre de cette valeur initiale entraîne, lors de la première utilisation, un message d'avertissement à l'exécution SMSL (si les avertissements à l'exécution sont activés). Il est donc préférable d'attribuer initialement une valeur de "" ou 0 à une variable liste/chaîne ou à une variable numérique, respectivement.

F.9 Constantes SMSL prédéfinies

Un certain nombre d'identificateurs sont prédéfinis comme constantes de manière à pouvoir être utilisés sans qu'une déclaration ne soit nécessaire. Ces constantes ne sont pas modifiables et il convient de ne pas les attribuer.

Tableau F.3 – Constantes SMSL prédéfinies

Constante	Définition
ALARME	état d'objet ALARME
OK	état d'objet OK
OFFLINE	état d'objet OFFLINE
VOID	état d'objet VOID
EOF	End-of-file condition constant
vrai/VRAI/Vrai oui/OUI/Oui	Valeur booléenne vraie (1)
faux/FAUX/Faux non/NON/Non	Valeur booléenne fausse (0)

F.10 Littéraux de chaîne SMSL

Les littéraux de chaîne sont délimités par des guillemets. Les littéraux de chaîne peuvent être constitués de plusieurs lignes de telle sorte que les caractères d'interligne font partie de la chaîne.

Les règles de barre oblique inverse s'appliquent pour les caractères d'échappement (tels que la barre oblique inverse ou les guillemets) ainsi que pour la construction de caractères tels que les caractères d'interligne ou de tabulation. Ce sont les seuls littéraux de chaîne actuellement pris en charge par le langage SMSL.

Tableau F.4 – Littéraux de chaîne SMSL

Constante	Définition
\t	tabulation
\n	caractère d'interligne
\r	retour
\b	caractère d'espacement arrière
\A . . . \Z	Ctrl-A . . . Ctrl-Z

Les caractères de commande peuvent être intégrés dans les constantes de chaîne SMSL en utilisant \A à \Z pour représenter Ctrl-A à Ctrl-Z. Une lettre en majuscule doit toujours être utilisée; les lettres minuscules autres que celles qui sont déjà définies (c'est-à-dire des t, n, r ou b) ne sont pas valables en tant que caractères d'échappement et donneront lieu à un avertissement de compilation SMSL.

F.11 Listes SMSL

Les valeurs de liste sont représentées en séparant les valeurs unitaires par des virgules et en délimitant la liste par des crochets:

[1,3,5]

La liste est interpolée en une chaîne entre guillemets dont les éléments sont séparés par des espaces. La liste est représentée en interne par

"1 3 5"

F.12 Instructions SMSL simples

L'instruction simple la plus courante est une expression évaluée pour ses effets secondaires, appelée instruction d'expression. L'instruction d'expression la plus courante est une opération d'affectation ou un appel de fonction. Chaque instruction d'expression doit se terminer par un point-virgule:

```
y = x + 10;          N° d'affectation

set("value",50);     N° d'appel de fonction

s = trim(s,"\t");    N° d'affectation et d'appel de fonction
```

F.13 Opérateurs SMSL

F.13.1 Opérateurs arithmétiques

En ce qui concerne les opérateurs arithmétiques, un opérande est considéré comme un nombre si son premier caractère est un chiffre ou un signe moins (–). Sinon, il est considéré comme une chaîne et est converti à 0 pour une chaîne vide ou à 1 pour une chaîne non vide.

L'utilisation d'une lettre non numérique dans un contexte arithmétique peut entraîner un avertissement à l'exécution.

Tableau F.5 – Opérateurs arithmétiques SMSL

Opérateur	Définition
+	addition
–	soustraction
/	division
*	multiplication
%	module

F.13.2 Opérateurs d'affectation

Les opérateurs d'affectation pour SMSL sont les suivants.

Par exemple, a+=b est équivalent à a=a+b.

Tableau F.6 – Opérateurs d'affectation SMSL

Opérateur	Définition
=	affectation
.=	autoconcaténation des chaînes
+=	auto-addition
–=	autosoustraction
/=	autodivision
*=	automultiplication
%=	automodule

Attribution au niveau binaire:

&=	autobinaire ET
|=	autobinaire OU
^=	attribution binaire OU exclusif

Attribution de déplacement:

<<=	attribution de déplacement à gauche
>>=	attribution de déplacement à droite

Opérateurs d'incrémentation/décrémentation

Par exemple, a++ est équivalent à a=a+1.

Tableau F.7 – Opérateurs incrémentation/décrémentation SMSL

Opérateur	Définition
++	incrément
--	décrément

F.13.3 Opérateurs au niveau binaire

Les opérateurs au niveau binaire pour SMSL sont les suivants.

Par exemple, a&=b est équivalent à a=a&b.

Tableau F.8 – Opérateurs SMSL au niveau binaire

Opérateur	Définition
&	au niveau binaire ET
|	au niveau binaire OU
&=	autobinaire ET
|=	autobinaire OU
^	au niveau binaire OU exclusif
^=	attribution au niveau binaire OU exclusif

F.13.4 Opérateurs logiques

Les opérateurs logiques SMSL considèrent que vrai est représenté par 1 ou par une chaîne non vide pour leurs opérandes. Faux est représenté par 0 ou par une chaîne vide. Cependant, lorsqu'ils renvoient des résultats, ils utilisent toujours 1 pour vrai et 0 pour faux.

Tableau F.9 – Opérateurs logiques SMSL

Opérateur	Définition
&&	ET logique
||	OU logique
!	NON logique

F.13.5 Opérateurs de relation

Les opérateurs de relation effectuent des comparaisons numériques si les deux opérandes sont des nombres. Sinon, ils exécutent des comparaisons de chaînes (c'est-à-dire lexique, réorganisation dictionnaire). Une chaîne est considérée comme un nombre si elle n'est composée que de chiffres, du signe moins ou d'un point. Les espaces ne sont pas autorisés. Les opérateurs de relation SMSL ne considèrent pas les constantes dans la notation exponentielle (telle que 2.3e+27) comme des nombres.

F.13.6 Opérateurs de déplacement

Les opérateurs de déplacement effectuent des déplacements binaires au niveau des octets.

Tableau F.10 – Opérateurs de relation SMSL

Opérateur	Définition
==	égal à
!=	non égal à
<	inférieur à
>	supérieur à
<=	inférieur ou égal à
>=	supérieur ou égal à

Tableau F.11 – Opérateurs de déplacement SMSL

Opérateur	Définition
<<	déplacement à gauche
<<=	attribution de déplacement à gauche
>>	déplacement à droite
>>=	attribution de déplacement à droite

F.13.7 Opérateurs de chaîne

SMSL dispose de plusieurs opérateurs pour la manipulation des chaînes et des listes.

[s1, s2, ...]

L'opérateur de liste construit une liste en rassemblant tous les éléments dans une liste séparée par une virgule dans une chaîne d'éléments entre guillemets délimitée par un espace, ce qui constitue la représentation de SMSL pour les listes/matricielles.

=~ (égal tilde)

L'opérateur =~ est utilisé dans la chaîne d'expression =~ modèle et retours:

- 1 si le modèle d'expression normale est contenu dans la chaîne;
- 0 si le modèle d'expression normale n'est pas contenu dans la chaîne.

Si le modèle est non valide, SMSL renvoie un message d'erreur à l'exécution et l'opération =~ renvoie 0 (modèle non contenu).

!~ (tilde)

L'opérateur !~ est utilisé dans la chaîne d'expression !~ modèle et retours:

- 1 si le modèle d'expression normale n'est pas contenu dans la chaîne;
- 0 si le modèle d'expression normale est contenu dans la chaîne.

Si le modèle est non valide, SMSL renvoie un message d'erreur à l'exécution et l'opération !~ revient à 0 (modèle contenu).

F.13.8 Priorité et associativité des opérateurs SMSL

La priorité et l'associativité des opérateurs SMSL sont pratiquement identiques à celles de C et de Perl. Outre les opérateurs standards, il existe de nouveaux opérateurs de chaîne – "." et [x,y,...] – avec leurs priorités associées.

Dans le Tableau F.12, les opérateurs sont inscrits en ordre croissant de priorité:

Tableau F.12 – Priorité et associativité des opérateurs SMSL

Priorité des opérateurs	Associativité
=	inférieure droite
+=, –=, <<=, >>=, ^=	droite
*=, /=, %=	droite
|=, &=	droite
||	gauche
&&	gauche
|	gauche
^	gauche
&	gauche
!=, ==, =~, !~	gauche
<, <=, >, >=	gauche
<<, >>	gauche
+, – (binaire)	gauche
*, /, %	gauche
. (concat chaîne)	gauche
!, –, ++, --	droite
()	gauche
[] supérieur	gauche

F.14 Langage de description central SMSL

Un script SMSL est constitué d'une séquence de commandes. Tous les objets non initialisés créés par l'utilisateur sont supposés commencer par une valeur NULL ou 0 jusqu'à ce qu'ils soient définis par une quelconque opération explicite, telle qu'une attribution.

Le langage SMSL est, en général, un langage à structure non imposée. C'est-à-dire que les lignes ne doivent pas commencer ou finir au niveau ou avant une colonne particulière; elles peuvent continuer sur la ligne suivante. On ne tient pas compte des espaces, sauf pour la séparation des jetons. Les commentaires sont indiqués par le numéro de caractère (#) et continuent jusqu'à la fin de la ligne. Par exemple, voici un commentaire sur une instruction d'affectation:

```
x = y;          # Affecter la valeur de y à la variable x
```

F.14.1 Instructions composées SMSL

Les instructions composées SMSL comprennent des instructions loop (boucle) et if (si). Dans le langage SMSL, une série d'instructions peut être traitée comme une seule instruction en la mettant entre accolades {}. Nous l'appellerons bloc d'instruction et nous le désignerons dans les descriptions d'instruction comme {BLOC}.

F.14.1.1 Exit (quitte)

Format

exit

Description

L'instruction exit donne lieu à la fin immédiate du programme SMSL qui redonne le contrôle au processus qui l'a appelé. Cette instruction doit se terminer par un point-virgule lorsqu'elle est utilisée dans un programme SMSL.

F.14.1.2 Export

Format

export variable

export function function

Paramètres

Description

L'instruction export permet d'exporter une variable ou une fonction d'une bibliothèque SMSL vers une autre bibliothèque ou un autre programme SMSL en utilisant l'instruction "requires". Chaque instruction d'export peut spécifier une seule variable ou fonction.

Les variables globales et les fonctions ne doivent pas être déclarées avant l'instruction export. L'instruction export n'exige pas qu'une variable soit définie explicitement dans une bibliothèque, mais qu'elle apparaisse dans une instruction SMSL afin de créer une définition implicite.

Position de l'instruction export

L'instruction "export function function" peut apparaître avant ou après la définition de fonction réelle. L'instruction "export variable" peut apparaître avant ou après la première apparition d'une variable globale.

Une instruction export peut apparaître dans une définition de fonction sans avoir une signification particulière.

Définition des paramètres

nom de variable d'une variable SMSL qui est disponible pour l'exportation vers un autre programme SMSL

nom de fonction d'une fonction SMSL qui est disponible pour l'exportation vers un autre programme SMSL

Erreurs liées à l'instruction export

L'instruction export peut générer des erreurs de compilateur dans les instances suivantes:

- la variable ou la fonction n'est pas définie ou utilisée dans la bibliothèque;
- la variable ou la fonction est une fonction intégrée SMSL;
- la variable est une variable locale d'une fonction définie par un utilisateur dans la bibliothèque;
- la variable ou la fonction est dupliquée dans une autre instruction export;
- la variable ou la fonction a été importée en utilisant l'instruction "requires".

F.14.1.3 Foreach

Format

foreach [list] {BLOCK}

foreach unit variable [list] {BLOCK}

Description

La boucle "foreach" répète sur des listes et sur des ensembles une variable qui va sur chaque élément de la liste, en réalisant des BLOCS pour chaque élément de la liste, tour à tour.

Définition des paramètres

liste: Une liste qui contient un ou plusieurs éléments qui peuvent correspondre à la variable.

BLOC: Une ou plusieurs instructions qui sont exécutées lorsqu'une variable correspond à un élément de la liste.

l'unité contrôle la manière dont la liste est divisée en éléments individuels.

Plage valide:

- le mot suppose que les éléments matriciels sont séparés par des espaces (espaces, tabulations, ou \n);
- la ligne suppose que les éléments matriciels sont séparés par \n.

par défaut si non spécifié: ligne

variable le nom de l'élément qui correspond à chaque élément dans la liste.

Exemples

Les exemples suivants soulignent l'usage de l'instruction "foreach".

Grouper les éléments dans une matrice

```
somme = 0;
élément foreach ("1\n2\n3\n4\n5")
{
        somme += élém;
}
```

Inscrire l'ID d'ouverture de session pour chaque compte sur le système

```
utilisateur foreach (cat ("/etc/passwd"))
{
        printf(ntharg (item, 1, ":"), "\n");
}
```

NOTE – cat() et ntharg() sont des fonctions intégrées SMSL.

Compter le nombre de mots dans une chaîne

```
mots = 0;
foreach mot w ("The cat sat on the mat.")
{
        mots++;
}
```

F.14.1.4 Fonction

Format

```
function name(argument_list) {BLOCK}
```

Description

L'instruction de fonction fournit des fonctions définies par l'utilisateur au sein des programmes SMSL semblables à celles disponibles dans le langage de programmation C. Le mot clé de fonction est nécessaire dans une définition de fonction d'utilisateur. Deux mots clés, local et return, sont optionnels:

- local déclare des variables qui seront utilisées uniquement au sein de la fonction;
- return identifie le résultat de la fonction qui est renvoyé à l'appelant.

Les fonctions doivent être définies avant d'être utilisées pour la première fois et la liste d'arguments correcte doit être transférée par un appel de fonction. Un appel de fonction renvoie toujours une chaîne de caractères représentant une chaîne de caractères ou une valeur numérique. (Tous les types de données sont représentés au sein de SMSL en tant que chaînes de caractères.)

Définition des paramètres

nom: L'étiquette de caractère utilisée pour identifier et appeler la fonction du programme SMSL; le nom ne peut pas être identique à:

- une fonction interne de SMSL intégrée;
- une variable SMSL.

liste d'arguments: Zéro ou plusieurs variables SMSL qui sont transférées à la fonction en tant que paramètres lorsque l'exécution est lancée. La liste d'arguments peut être une entrée NULL si aucune variable n'est transférée à la fonction, un argument ou plusieurs arguments séparés par des virgules.

BLOC: Une ou plusieurs instructions SMSL qui définissent l'action que la fonction exécute.

Les arguments sont transférés par valeurs aux paramètres (c'est-à-dire que des copies locales des données d'arguments transférées sont créées) et, par conséquent, la modification d'un paramètre n'affectera pas la valeur de l'argument. Les paramètres de fonction correspondent à une fonction locale et peuvent avoir les mêmes noms que les variables globales (ou que les paramètres d'autres fonctions).

Si une définition de fonction apparaît au milieu des instructions exécutables et le flux de commande atteint la définition ci-dessus, on saute la définition comme si c'était un commentaire. La seule manière d'accéder à une fonction est de l'appeler explicitement. Les définitions de fonctions ne servent qu'à définir une fonction et ne sont disponibles que lorsqu'elles sont appelées. Il est donc possible de placer le code exécutable au-dessus, en dessous et entre les définitions de fonction.

F.14.1.5 Instruction RETURN

Il existe trois moyens de quitter une fonction définie par un utilisateur:

- return avec une valeur return;
- return sans une valeur return (valeur return = chaîne NULL);
- passer directement à la dernière accolade à droite (return omis, pas de valeur return).

SMSL n'interprète pas le fait de passer directement à la fin d'une fonction comme une condition d'erreur.

SMSL génère un avertissement de compilation semblable à celui des compilateurs C lorsqu'il rencontre des instructions return, certaines avec des valeurs return et d'autres sans, au sein d'une fonction. Le fait de disposer de plusieurs points de sortie permettant de quitter une fonction de différentes manières peut prêter à confusion quant à savoir si la fonction a été définie pour exécuter une action ou renvoyer une valeur.

F.14.2 Définition des variables locales

Des variables locales de fonction définies par des utilisateurs sont déclarées en utilisant le mot clé local au sein du corps de la fonction. Le mot clé local déclare une ou plusieurs variables spécifiées dans une liste séparée par une virgule qui se termine par un point-virgule. Ces noms deviennent des variables locales pour la fonction. Exemple de définitions de variables locales:

```
function f()
{
        local x;
        local a,b;
        # ... Statements for the function execution.
}
```

Les variables locales ne peuvent pas avoir le même nom qu'un paramètre de fonction ou qu'une autre variable locale dans la même fonction. Les noms de variables locales au sein d'une fonction n'influencent pas ceux d'une autre fonction. Les variables locales peuvent avoir le même nom qu'une variable globale et peuvent ainsi "cacher" un nom global.

Les déclarations de variable locale sont traitées comme des expressions et peuvent apparaître n'importe où au sein de la fonction pourvu qu'une expression soit valide. Cependant, il n'y a aucun concept de portée interne au sein des blocs intérieurs et la portée d'une variable locale s'étend de son point de déclaration jusqu'à la fin de la fonction dans laquelle elle est délimitée (pas le bloc de délimitation).

Les variables locales sont initialisées à la chaîne vide chaque fois que la fonction est saisie. Elles ne conservent pas les valeurs d'un appel précédent.

Le nombre maximal de variables locales et de paramètres de fonction dans des fonctions définies par l'utilisateur (à l'exception de la fonction main()-principale) est spécifique à l'application.

F.14.3 Fonction point d'entrée

La fonction point d'entrée SMSL est la fonction main()-principale. Si un programme SMSL contient une fonction définie par l'utilisateur nommée main, l'exécution commence à la première instruction de main(). Le programme SMSL se termine normalement au retour de main(). Il est admis que la fonction spécifiée comme le point d'entrée ait les mêmes propriétés que main().

La fonction main() ou la fonction point d'entrée doit être définie en tête du programme SMSL et non pas dans des bibliothèques importées. On ne tient pas compte des fonctions importées des bibliothèques lorsqu'on détermine si une fonction point d'entrée est disponible.

F.14.3.1 Commencer une exécution sans une fonction point d'entrée

Lorsque aucune fonction main() ou fonction point d'entrée n'est spécifiée en utilisant l'option de compilation SMSL, l'exécution commence à la première instruction exécutable qui n'est pas dans une définition de fonction. Dans un programme sans fonction d'entrée, les définitions de fonctions seront généralement placées en tête (elles doivent être définies avant leur première utilisation) suivies des instructions exécutables principales. Un exemple type est illustré ci-dessous:

```
function max(x,y)
{
        if(x > y)
        {
            return x;
        }
        else
        {
            return y;
        }
}
m = max(1,2);               # Execution starts here
printf("maximum is", m, "\n");
```

Comme on peut le constater, l'exécution du programme commence immédiatement après la définition de la fonction.

F.14.3.2 Limites des fonctions définies par l'utilisateur

Les fonctions définies par l'utilisateur sont soumises aux limitations suivantes:

- les appels de fonctions sont non récurrents;
- les fonctions définies par l'utilisateur peuvent faire des appels illimités aux autres fonctions pourvu qu'il n'y ait pas de récurrence directe ou indirecte dans la séquence des appels;
- transfert par valeur et par référence pris en charge;
- les fonctions SMSL prennent en charge les arguments transférés par référence et par valeur.

Limites de paramètres et de variables locales

Les limites de paramètres et de variables locales pour les fonctions SMSL ne sont pas définies dans la présente Spécification. Elles sont spécifiques à l'application.

Imbrication de fonctions non autorisée

SMSL n'autorise pas l'imbrication de fonctions – chaque définition de fonction doit être de portée globale et ne peut être définie au sein d'une autre fonction.

F.14.4 If

Format

if (expression) {BLOCK}

if (expression) {BLOCK} else {BLOCK}

if (expression) {BLOCK} elsif (expression) {BLOCK} . . . else {BLOCK}

Description

L'instruction if est directe. Du fait qu'une instruction de BLOC est toujours limitée par des accolades, il n'y a aucune ambiguïté quant à la relation avec if, elsif et else.

Exemples

Les exemples suivants soulignent l'usage de if, elsif, et else.

Une instruction if

```
if (x > 10)
{
        x = 10;             # don’t let x get bigger than 10
}
```

Définition des paramètres

expression: Instruction SMSL pour laquelle l'évaluation retourne à VRAI ou FAUX.

BLOC: Une ou plusieurs instructions SMSL qui sont exécutées une fois selon l'évaluation des expressions if ou elsif.

Description

L'instruction if est directe. Du fait qu'une instruction BLOC est toujours limitée par des accolades, il n'y a aucune ambiguïté quant à la relation avec if, elsif, et else.

Exemples

Les exemples suivants soulignent l'usage de if, elsif, et else.

```
Une instruction if
if (x > 10)
{
       x = 10;     # don’t let x get bigger than 10
}
Une instruction if . . . else
if (x == 0)
{
       # do something
}
```

```
else
{
       # x != 0
       # do something else
}
Une instruction if . . . elsif . . . else
if (x == 0)
{
       # do something
}
elsif (x == 1)
{
       # do something else
}
else
{
       # x !== 0 &&  x != 1
       # do something else
}
```

F.14.5 Last

Format

last

Description

L'instruction last indique au programme SMSL de quitter la boucle d'exécution la plus à l'intérieur. L'instruction last doit se terminer par un point-virgule lorsqu'elle est utilisée dans un programme SMSL.

F.14.6 Next

Format

next

Description

L'instruction next démarre immédiatement la prochaine itération de la boucle d'exécution la plus à l'intérieur.

F.14.7 Requires

Format

requires library

Description

L'instruction requires importe des variables et des fonctions identifiées dans des instructions export d'une bibliothèque SMSL déjà créée dans le programme SMSL. Chaque instruction requires peut spécifier un nom de bibliothèque.

SMSL ne contient pas d'instruction import explicite; l'utilisation de l'instruction requires implique importation. L'instruction requires recherche le binaire contenant la bibliothèque, lit toutes ses informations concernant l'instruction export et importe les variables et/ou fonctions spécifiées dans le programme SMSL.

Un nombre illimité d'instructions requires peut apparaître dans un programme SMSL. Toutes les bibliothèques spécifiées dans les instructions requires doivent être disponibles au compilateur lors de la compilation.

Instructions requires au sein de bibliothèques importées

Le compilateur SMSL résoudra automatiquement les dépendances imbriquées dans des bibliothèques importées, mais ne chargera pas automatiquement toutes les autres fonctions et variables exportées contenues dans la bibliothèque qui correspond à la dépendance imbriquée. Il faut importer une bibliothèque explicitement afin d'avoir accès à toutes les variables et fonctions exportées qui s'y trouvent.

Définition des paramètres

bibliothèque: Nom de la bibliothèque pour laquelle les variables et fonctions d'export spécifiées sont à importer dans le programme SMSL.

Une instruction requires peut apparaître à l'intérieur d'une définition de fonction sans avoir une signification particulière.

Disponibilité des variables et des fonctions au sein des bibliothèques importées

Lorsqu'un programme SMSL importe des variables et des fonctions de plusieurs bibliothèques, les variables et fonctions importées d'une bibliothèque peuvent modifier et utiliser les variables et fonctions importées des autres bibliothèques, sans considérer la manière dont les bibliothèques sont chargées pour la compilation.

Erreurs liées à l'instruction requires

L'instruction requires peut générer des erreurs de compilateur dans les instances suivantes:

- Une référence à une variable ou à une fonction importée apparaît devant l'instruction requires qui l'importe. Il faut mettre une instruction requires avant la première utilisation de la variable ou de la fonction importée.
- Une fonction importée a le même nom qu'une fonction définie dans le programme SMSL.
- Le même nom de variable ou de fonction est importé de deux bibliothèques ou plus.

F.14.8 Switch

Format

```
switch (variable)
{
    case a: {BLOCK}
    case b: {BLOCK}
    . . .
    case p,q,r: {BLOCK}
    . . .
    case n: {BLOCK}
    default: {BLOCK}
}
```

Paramètres

Description

L'instruction switch évalue la variable et exécute un BLOC SMSL spécifique par rapport à la valeur de son entier. Les étiquettes case correspondent aux valeurs de la variable pour laquelle un BLOC SMSL spécifique est disponible.

Si la valeur de la variable se situe en dehors de la plage des valeurs des étiquettes de cas, l'exécution continue avec le BLOC correspondant à l'étiquette par défaut. Si aucune étiquette par défaut n'existe, l'exécution continuera avec la première instruction suivant l'instruction switch.

Définition des paramètres

variable: Nom de variable SMSL dont l'entier spécifie le BLOC d'instruction SMSL qui sera exécuté.

a,b, . . . p,q,r, . . . n Valeurs de l'entier indiquant la valeur de la variable qui donnera lieu à l'exécution du BLOC correspondant.

BLOC: Une ou plusieurs instructions qui sont exécutées lorsque la valeur du cas correspondant est égale à la variable.

Description

L'instruction switch évalue la variable et exécute un BLOC SMSL spécifique par rapport à la valeur de son entier. Les étiquettes de cas correspondent aux valeurs de la variable pour laquelle un BLOC SMSL spécifique est disponible.

Si la valeur de la variable se situe en dehors de la plage des valeurs des étiquettes de cas, l'exécution continue avec le BLOC correspondant à l'étiquette par défaut. Si aucune étiquette par défaut n'existe, l'exécution continuera avec la première instruction suivant l'instruction switch.

L'instruction switch SMSL se comporte de la même manière qu'une longue séquence d'instructions if-then-else-if. Un cas ou une clause par défaut est en effet une instruction à l'exécution qui spécifie une comparaison par rapport à la valeur d'une variable:

- Si la valeur de la variable correspond à un cas, l'exécution a lieu au sein du BLOC pour le cas ou la clause par défaut; et une fois le BLOC terminé, l'exécution continue après l'instruction switch entière (c'est-à-dire qu'il n'y pas de passage direct à la prochaine clause de cas).
- Si la valeur de la variable ne correspond pas à un cas, l'exécution saute à la clause par défaut; si aucune clause par défaut n'existe, l'exécution a lieu à l'instruction suivant l'instruction switch.

Toute instruction, dans le bloc du cas d'instruction switch qui ne fait pas partie d'un cas ou BLOC par défaut, n'a lieu que si toutes les étiquettes de cas précédentes n'ont pas pu correspondre à la variable (c'est-à-dire qu'elle est exécutée comme partie de la séquence normale du flux de commande).

Les propriétés de l'instruction switch SMSL sont les suivantes:

- Les expressions de cas peuvent être des expression évaluées dynamiquement ainsi que des expressions constantes.
- Le séparateur "deux-points" qui sépare l'étiquette de cas du BLOC exécutable est optionnel dans le SMSL.
- Le SMSL exige que l'étiquette par défaut suive toutes les étiquettes de cas dans le bloc cas de l'instruction switch. Il renvoie une erreur de compilation si une ou plusieurs étiquettes de cas suivent l'étiquette par défaut.
- Le SMSL ne renvoie pas une erreur de compilation pour des étiquettes de cas dupliquées dans l'instruction switch. Dans le SMSL, on ne peut atteindre la deuxième étiquette de cas dupliquée.
- Le SMSL permet de spécifier plusieurs cas qui exécutent un BLOC commun comme une liste séparée par une virgule à l'intérieur d'une seule étiquette de cas. (Réciproquement, les étiquettes empilées ne fonctionneront pas dans SMSL.)
- L'exécution d'un BLOC SMSL ne passe pas directement à l'étiquette de cas et au BLOC suivant. Lorsqu'elle atteint l'accolade de fermeture d'un cas ou d'un BLOC par défaut, l'exécution se déplace vers la fin de l'instruction switch SMSL.
- L'instruction switch SMSL utilise la dernière instruction pour quitter un BLOC. La dernière instruction permet de quitter l'instruction switch ou la boucle la plus à l'intérieur. Du fait de l'absence de "transfert" dans SMSL, il n'est presque pas nécessaire d'utiliser la dernière instruction dans l'instruction switch.
- Le SMSL génère une erreur de compilateur lorsqu'il détecte deux étiquettes par défaut dans une seule instruction switch.
- Le SMSL autorise des instructions switch imbriquées.

Les BLOCS de cas sont évalués à l'exécution par leur ordre d'apparition:

- par ordre de cas pour les BLOCS;
- de gauche à droite pour les expressions contenues dans les listes séparées par des virgules des étiquettes de cas multiples.

Toutes les expressions d'une liste séparée par des virgules sont évaluées avant l'étiquette de cas. Cette évaluation a lieu même si la première expression correspond.

Cette séquence et cette méthode d'évaluation de l'étiquette de cas peuvent entraîner des risques si une expression de la liste modifie la variable pour l'instruction switch en cours ou une variable utilisée dans une autre expression de cas. Sous SMSL, les instructions au sein d'une instruction switch qui ne font pas partie d'un BLOC (instructions libres) peuvent et seront exécutées si elles sont atteintes par le flux d'exécution. Pour que le flux de commande atteigne ces

instructions, les variables ne peuvent correspondre à aucune des étiquettes de cas qui les précède à l'intérieur de l'instruction switch. Le SMSL ne renvoie pas un avertissement ou un message d'erreur lorsque deux étiquettes de cas évaluées par rapport à une variable sont imbriquées l'une dans l'autre. Deux exemples de cette situation sont illustrés dans l'exemple de switch SMSL suivant:

switch(x) { case 1: { f1() # Function f1 Called case 2: {f2();} # Function f2 Unreachable f3(); # Function f3 Called } default: {case 4: {f4();}} # Function f4 called if x=4 }

Dans la mesure où les étiquettes de cas et par défaut sont des instructions à l'exécution, l'effet d'une étiquette de cas imbriquée dans une autre exige que cette variable corresponde à la valeur de cas pour le BLOC de cas à exécuter. Ceci implique qu'une variable doit être égale à deux valeurs différentes! Dans le cas 1 de l'exemple, f2 ne sera jamais appelé parce que x ne peut pas être égale à 1 et à 2.

Dans le cas par défaut de l'exemple, f4 sera appelé si la variable = 4 parce qu'il n'y a pas de cas 4 défini dans l'instruction switch. Lorsque la variable = 4, le BLOC par défaut contenant le BLOC de cas 4 d'appel à la fonction f4, est exécuté.

F.14.9 While

Format

while (expression) {BLOCK}

Paramètres

Description

La boucle while exécute des instructions tant que l'expression donne VRAI comme résultat (non nul).

Exemple

Les exemples d'instructions SMSL suivants affichent les entiers de 1 à 10.

```
x = 1;
while (x <= 10)
{
        printf (x, " ");
        x++;
}
printf ("\n");
```

Définition des paramètres

expression: Une instruction SMSL dont l'évaluation renvoie VRAI ou FAUX.

BLOC: Une ou plusieurs instructions SMSL qui exécutent répétitivement tant que l'expression donne VRAI.

Description

La boucle while exécute des instructions tant que l'expression donne VRAI (non nul).

Exemple

Les exemples d'instructions SMSL suivants affichent les entiers de 1 à 10.

```
x = 1;
while (x <= 10)
{
        printf (x, " ");
        x++;
}
printf("\n");
```

F.14.10 Modèle objet

Le SMSL est fondé sur un modèle orienté objet simple qui rend possible une mise en correspondance à partir de l'environnement de gestion-systèmes décrit dans les GDMO (directives pour la définition des objets gérés) (ISO/CEI 10164-4). Un objet est une construction dont les propriétés sont des variables ou d'autres objets. Les fonctions associées à un objet sont les méthodes de l'objet.

Un utilisateur SMSL peut accéder aux propriétés d'un objet avec la notation suivante:

objectName.propertyName

Cet objet peut être un objet GDMO ou un quelconque objet défini localement.

Si cet objet est un objet GDMO, la mise en correspondance des propriétés GDMO et des propriétés SMSL est illustrée dans le tableau suivant:

Tableau F.13 – Mise en correspondance entre GDMO et SMSL

Propriété GDMO	Propriété SMSL
étiquette de classe	nom du type d'objet
identificateur d'instance (ENTIER, etc.)	valeur de la variable d'instance d'objet
valeurs initiales lors de la création d'une instance	paramètres de la "nouvelle" opération
noms d'instance de l'objet géré	notation de valeur ASN.1
type d'ATTRIBUTS ASN.1	nom des variables
étiquette des GROUPES D'ATTRIBUTS	nom de la variable matricielle
étiquette de l'ACTION	nom de la méthode (fonction)
traitement de NOTIFICATION asynchrone	programme onEvent(discriminatorConstruct)

Une propriété peut être définie en lui affectant une valeur comme suit:

objectName.propertyName = value;

Une méthode est une fonction associée à un objet. Une fonction peut être associée à un objet comme:

objectName.methodName = functionName

où l'objet est un objet local existant, la méthode est le nom affecté à la méthode et functionName est le nom de la fonction.

La méthode dans le contexte de l'objet peut être appelée telle que suit:

objectName.methodName(parameters);

F.14.10.1 Opération GDMO et paramètres d'action

Les paramètres d'opération et d'action sont transférés à la méthode d'action comme un objet mis en correspondance par les règles décrites dans le modèle objet. Les valeurs de paramètres de retour sont renvoyées de la méthode d'action (fonction) comme un objet mis en correspondance par ces règles. Le format de description est le suivant:

outputObjectName = ObjectName.actionName(inputObjectName);

outputObjectName = ObjectName.operationName(inputObjectName);

F.14.10.2 Référence d'objet "this"

SMSL dispose d'un mot clé spécial, "this", qui peut être utilisé pour faire référence à l'objet courant.

this[propertyName]

F.14.10.3 Création et suppression des objets

new

Un opérateur qui vous permet de créer une instance d'un type d'objet défini par l'utilisateur.

La création d'un type d'objet exige deux étapes:

1) Définir le type d'objet en écrivant une fonction.

2) Créer une instance de l'objet avec "new".

Pour définir un type d'objet, créer une fonction pour le type d'objet qui spécifie son nom, ses propriétés et ses méthodes. Un objet peut avoir une propriété qui est elle-même un autre objet.

Format

objectName = new objectType (param1 [,param2] .... [,paramN])

objectName est le nom de l'instance d'objet new.

objectType est la fonction qui définit un type d'objet.

param1..paramN sont des valeurs de propriété pour l'objet. Ces propriétés sont des paramètres définis pour la fonction objectType.

L'instance d'une classe peut être supprimée avec l'opérateur "delete".

delete objectName

objectName est le nom de l'instance d'objet existant.

Par ailleurs, le type d'objet local peut être défini en écrivant une fonction. Une instance locale de l'objet peut ensuite être créée avec l'opérateur "new".

Expression d'objet pour représenter la corrélation de noms

Si deux objets ont une relation de corrélation de noms, l'expression suivante de l'objet subordonné est autorisée:

superiorObjectInstance.subordinateObjectInstance

F.14.10.4 WITH

Une instruction qui établit un objet par défaut pour un ensemble d'instructions. Dans l'ensemble d'instructions, toute référence à la propriété qui ne spécifie pas un objet est supposée être pour l'objet par défaut.

Syntaxe

with (*objectName*){

statements

}

objectName spécifie l'objet par défaut à utiliser pour les *instructions*. Il faut mettre *objectName* entre parenthèses. *statements* est n'importe quel bloc d'instructions.

F.14.10.5 Programme Event

onEvent

Description

Un opérateur de traitement d'événement pour décrire le traitement effectué quand une notification spécifiée définie par les GDMO a lieu.

Syntaxe

onEvent(*DiscriminatorConstructValue*){

statements

}

DiscriminatorConstructValue est la valeur type DiscriminatorConstruct. *statements* est n'importe quel bloc d'instructions.

F.14.10.6 triggerParameterCount

Cette machine virtuelle SMSL compte le nombre de paramètres de déclenchement au moyen de cette variable.

F.14.10.7 triggerArgument

Description

Le triggerArgument accepte la liste de paramètres de déclenchement accessible depuis un script SMSL et renvoie l'identité du pas de lancement qui exécute ensuite le script. La liste d'arguments comprend l'id du déclencheur et l'id du script sauf si le déclencheur n'est pas paramétré.

Syntaxe

triggerArgument(*argument_list*), where *argument_list* can have up to *triggerParameterCount* elements.

Exemple d'un script SMSL pour la gestion de EFD (discriminateur de retransmission d'événements)

Cet exemple décrit comment créer un EFD et déterminer son état opérationnel.

Les paramètres pour le script à déclencher sont l'id de script et la destination de la notification. Le script SMSL pour obtenir l'attribut de l'état opérationnel de l'EFD et le renvoyer comme une notification à la destination est cité ci-dessous.

Le déclencheur est spécifié avec les paramètres suivants:

- *triggerId*, l'identificateur de ce déclencheur;
- *scriptId*, l'identificateur du script à exécuter;
- *managerId*, l'identificateur de la destination à laquelle doivent être retransmis les rapports d'événements.

Le script SMSL, pour cet exemple, est cité ci-dessous:

```
#include "CMIP.CMIP-1.h"
#include "DMI.Attribute-ASN1Module.h"

manager1 = triggerArgument(triggerId, scriptId, managerId);

/* Instance creation as CMISFilter type */
CMISFilter counterValue _GT10 = item ( greaterOrEqual ( Attribute-ASN1Module.Count, 10))

/*
* Creating instance of EFD with the following parameters.
* DiscriminatorConstruct = ( counter > 10 ),
* administrativeState = default value, destination = manager1
*/

efd1 = new eventForwardingDiscriminator( counterValue_GT10, manager1);

triggerResult = efd1.operationalState;                    /* get operational state */

printf("operational state of event forwarding discriminator %d \n", triggerResult);
```

Le gestionnaire de destination est fourni comme un paramètre déclencheur. Le déclencheur force le pas de lancement à générer de manière dynamique des tâches élémentaires afin d'exécuter toutes les instructions de script séquentiellement. Le pas de lancement transfère les paramètres appropriés aux tâches élémentaires. Lors de la création de efd1, par exemple, le pas de lancement transfère l'id de script et le gestionnaire de destination comme des paramètres à la tâche élémentaire.

F.14.11 Description en BNF (formalisme de Backus Naur) du langage SMSL

(Conventions: "{}" désigne les productions qui apparaissent 0 fois ou plus; "[]" désigne les productions optionnelles.)

Jetons:

```
T_ELLIPSIS              : "..."
T_EQ                    : "=="
T_NE                    : "!="
T_REGEXP_EQ             : "=~"
T_REGEXP_NE             : "!~"
T_LEQ                   : "<="
```

```
T_GEQ                   : ">="
T_GT                    : ">"
T_LT                    : "<"
T_AND                   : "&&"
T_OR                    : "||"
T_NOT                   : "!"
T_INC                   : "++"
T_DEC                   : "--"
T_PLUSEQ                : "+="
T_MINUSEQ               : "-="
T_MULEQ                 : "*="
T_DIVEQ                 : "/="
T_MODEQ                 : "%="
T_BITANDEQ              : "&="
T_BITOREQ               : "|="
T_LEFT_SHIFT            : "<<"
T_RIGHT_SHIFT           : ">>"
T_RIGHT_SHIFT_ASSIGN    : ">>="
T_LEFT_SHIFT_ASSIGN     : "<<="
T_XOR_ASSIGN            : "^="
```

Mots clés:

```
T_IF                    : "if"|"IF"
T_ELSE                  : "else"|"ELSE"
T_ELSIF                 : "elsif"|"ELSEIF"
T_FOREACH               : "foreach"|"FOREACH"
T_FOR                   : "for"|"FOR"
T_WORD                  : "word"|"WORD"
T_LINE                  : "line"|"LINE"
T_NEXT                  : "next"|"NEXT"
T_LAST                  : "last"|"LAST"
T_WHILE                 : "while"|"WHILE"
T_DO                    : "do"|"DO"
T_UNTIL                 : "until"|"UNTIL"
T_EXIT                  : "exit"|"EXIT"
T_TRUE                  : "true"|"TRUE"|"True"|"yes"|"YES"|"Yes"
T_FALSE                 : "false"|"FALSE"|"False"|"no"|"NO"|"No"
T_RETURN                : "return"|"RETURN"
T_LOCAL                 : "local"|"LOCAL"
T_FUNCTION              : "function"|"FUNCTION"
```

```
T_NATIVE              : "native"|"NATIVE"
T_RPC                 : "rpc"|"RPC"
T_VOID                : "void"
T_REQUIRES            : "requires"|"REQUIRES"
T_EXPORT              : "export"|"EXPORT"
T_CASE                : "case"|"CASE"
T_SWITCH              : "switch"|"SWITCH"
T_DEFAULT             : "default"|"DEFAULT"
```

Règles:

```
program : stmts

stmts : { stmt }

stmt : expr ';'          |
       return    |
       if    |
       foreach    |
       switch     |
       case     |
       default    |
       while    |
       do_until  |
       for_loop  |
       T_LAST ';' |
       T_NEXT ';' |
       T_EXIT ';' |
       function   |
       native_function  |
       rpc      |
       local_var_dec    |
       export   |
       requires

requires : T_REQUIRES requires_name ';'

library_name : T_STRING | T_IDENTIFIER

requires_name : library_name
```

```
export : T_EXPORT [ T_FUNCTION ] export_name ';'

export_name : T_IDENTIFIER

part : T_WORD
foreach : T_FOREACH part simple_id '(' expr ')' '{' stmts '}'

case_exprs : { case_expr ',' } case_expr

case_expr : expr

case : T_CASE case_exprs optional_colon '{' stmts '}'

default : T_DEFAULT optional_colon '{' stmts '}'

optional_colon  :  [ ':' ]

switch : T_SWITCH '(' expr ')' '{' stmts '}'

for_loop : T_FOR '(' optional_expr ';' optional_expr ';' optional_expr ')'
          '{' stmts '}'

do_until : T_DO '{' stmts '}' T_UNTIL '(' expr ')' ';'

while : T_WHILE '(' expr ')' '{' stmts '}'

void : [ T_VOID ]

native_function :
       T_NATIVE void T_FUNCTION function_name '(' func_param_list ')' ';'

rpc : T_RPC void T_FUNCTION function_name '(' func_param_list ')' ';'

function : T_FUNCTION function_name '(' func_param_list ')' '{' stmts '}'

func_param_list : param_list

function_name : T_IDENTIFIER

param_list : [ { one_param ',' }  one_param ]
```

```
one_param : T_IDENTIFIER |
          T_ELLIPSIS

local_var_dec : T_LOCAL var_list ';'
var_list : { one_var ',' }  one_var

one_var : T_IDENTIFIER

return : T_RETURN [ expr ] ';'

if : T_IF '(' expr ')' '{' stmts '}' opt_elsifs opt_else

opt_elsifs : { elsif }

elsif : T_ELSIF '(' expr ')' '{' stmts '}'

opt_else : [ else ]

else : T_ELSE '{' stmts '}'

optional_expr : [ expr ]

expr : unary_expr             |
       expr '+' expr                |
       expr '-' expr                |
       expr '*' expr                |
       expr '/' expr                |
       expr '%' expr                |
       expr T_EQ expr               |
       expr T_NE expr               |
       expr T_REGEXP_NE expr        |
       expr T_REGEXP_EQ expr        |
       expr T_LT expr               |
       expr T_GT expr               |
       expr T_LEQ expr              |
       expr T_GEQ expr              |
       expr T_AND expr              |
       ternary_expr           |
       expr T_OR expr               |
       expr '|' expr                |
       expr '&' expr                |
       expr '^' expr                |
```

```
        expr T_LEFT_SHIFT expr       |
        expr T_RIGHT_SHIFT expr            |
        lvalue '=' expr              |
        lvalue T_PLUSEQ expr         |
        lvalue T_MINUSEQ expr        |
        lvalue T_MULEQ expr          |
        lvalue T_DIVEQ expr          |
        lvalue T_BITANDEQ expr       |
        lvalue T_BITOREQ expr        |
        lvalue T_MODEQ expr          |
        lvalue T_LEFT_SHIFT_ASSIGN expr    |
        lvalue T_XOR_ASSIGN expr           |
        lvalue T_RIGHT_SHIFT_ASSIGN expr   |
        expr '.' expr

simple_id : T_IDENTIFIER

ternary_expr : expr '?' expr ':' expr

lvalue : simple_id

unary_expr : primary          |
             '-' unary_expr   |
             T_NOT unary_expr |
             T_INC lvalue     |
             T_DEC lvalue

function_call_id : T_IDENTIFIER

primary : simple_id                      |
          T_INT                  |
          T_FLOAT                      |
          T_STRING                     |
          T_TRUE                 |
          T_FALSE                      |
          '(' expr ')'                 |
          lvalue T_INC                 |
          lvalue T_DEC                 |
          function_call_id '(' arglist ')'
```

```
arglist : [ { expr ’,’ } expr ]
=======================================================
```

A noter que la définition d'une chaîne doit permettre l'apparition d'un caractère \" dans la chaîne.

```
T_IDENTIFIER          : [A-Za-z_][A-Za-z_0-9]*
T_STRING              : \"[^"]*\"
T_FLOAT               : [0-9]*"."[0-9]+
T_INT                 : [0-9]+
```

/* Jetons d'opérateur et leurs priorités */

```
%right  ’=’  T_PLUSEQ  T_MINUSEQ  T_MULEQ  T_DIVEQ  T_MODEQ  T_BITANDEQ  T_BITOREQ
T_LEFT_SHIFT_ASSIGN T_RIGHT_SHIFT_ASSIGN T_XOR_ASSIGN
%left ’?’ ’:’
%left T_OR
%left T_AND
%left ’|’
%left ’^’
%left ’&’
%left T_EQ T_NE T_REGEXP_EQ T_REGEXP_NE
%left T_LT T_GT T_LEQ T_GEQ
%left T_LEFT_SHIFT T_RIGHT_SHIFT
%left ’+’ ’-’
%left ’*’ ’/’ ’%’
%left ’.’
%right T_UNARY T_NOT T_INC T_DEC
%left ’(’
%left ’[’
```

Annexe G

Fonctions support du langage SMSL

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

acos()

Renvoie l'arc cosinus de l'argument.

Format

acos(*cosine*)

Paramètre

Paramètre	Définition
cosine	argument cosinus domaine de validité: –1 <= *cosine* <= 1

Description

La fonction acos() renvoie l'arc cosinus de *cosinus*; c'est-à-dire, la longueur en radians de l'arc dont le cosinus est *cosine*.

La plage de résultat pour la fonctions acos() est $0 \leq \text{acos}() \leq \pi$. La valeur de retour de la fonction acos() est donnée avec une précision de 6 décimales.

asctime()

Renvoie la date et l'heure sous forme d'une chaîne de caractères.

Format

asctime(*clock,format*)

Paramètres

Paramètre	Définition
clock (*horloge*)	Référence à une horloge ou à un temporisateur dont il convient de convertir la valeur en une chaîne de caractères. *clock* est le plus souvent time().
format	Spécification facultative du format de la chaîne de sortie asctime(). Les spécificateurs de champ suivants sont valides: %a jour de la semaine abrégé %A jour de la semaine complet %b mois abrégé %B mois complet %c représentation locale de la date et de l'heure %d jour en décimal du mois (de 01 à 31) %E année et ère combinées %H heure en décimal en mode 24 heures (de 00 à 23) %I heure en décimal en mode 12 heures (de 01 à 12) %j jour en décimal de l'année (de 001 à 366) %m numéro du mois en décimal (de 01 à 12) %M minute en décimal (de 00 à 59) %n caractère d'interligne %N nom de l'ère/de la dynastie %o année %p équivalent de AM/PM

Paramètres
(fin)

Paramètre	Définition
format	Spécification facultative du format de la chaîne de sortie asctime(). Les spécificateurs de champ suivants sont valides: %S seconde en décimal (de 00 à 61) %t caractère de tabulation %U numéro de semaine en décimal de l'année: Dimanche est le premier jour de la semaine, tous les jours précédant le premier dimanche de l'année sont dans la semaine 0 (de 00 à 53) %w jour de la semaine en décimal: Dimanche est le premier jour de la semaine (de 00 à 06) %W numéro de semaine de l'année en décimal: Lundi est le premier jour de la semaine, tous les jours précédant le premier lundi de l'année sont dans la semaine 0 (de 00 à 53) %x représentation locale de la date %X représentation locale de l'heure %y y année en décimal sans partie séculaire (de 00 à 99) %Y année en décimal avec partie séculaire %Z nom du fuseau horaire (s'il existe) %% % caractère % Des spécificateurs de champ peuvent être exprimés sous la forme: [– | 0] *field_specifier*.p où – justifie à gauche le champ (justification à droite par défaut) 0 justifie à droite le champ et remplit avec des zéros à gauche .p nombre minimal de chiffres à afficher pour les champs numériques ou nombre maximal de caractères à afficher pour les champs alphabétiques. Pour les champs numériques, les positions vides sont remplies de zéros à gauche. Pour les champs alphabétiques, les caractères excédentaires sont tronqués à droite. Par défaut si non spécifié: chaîne de 24 caractères avec le format: Sun Sep 16 01:03:52 1973

Description

La fonction asctime() renvoie la date/heure de l'horloge sous la forme d'une chaîne de caractères. Elle équivaut à la fonction asctime() de la bibliothèque C.

Si un format est donné, asctime() renvoie la chaîne date/heure dans le format spécifié. Les spécificateurs de champs utilisés dans le format sont équivalents à ceux utilisés dans la fonction strftime() de la bibliothèque C.

asin()

Renvoie l'arc sinus de l'argument.

Format

asin(*sine*)

Paramètre

Paramètre	Définition
sine	Domaine de validité: $-1 \leq sine \leq 1$

Description

La fonction asin() renvoie l'arc sinus de *sine*; c'est-à-dire la longueur en radians de l'arc dont le sinus est *sine*. La plage de résultat pour la fonction asin() est $-\pi/2 \leq \text{asin}() \leq \pi/2$.

atan()

Renvoie l'arc tangente de l'argument.

Format

atan(*tangent*)

Paramètre

Paramètre	Définition
tangent	Domaine de validité: $-\infty \leq tangent \leq \infty$

Description

La fonction atan() renvoie l'arc tangente de *tangente*; c'est-à-dire la longueur en radians de l'arc dont la tangente est *tangent*.

La plage de résultat pour la fonction atan() est $-\pi/2 \leq \text{atan}() \leq \pi/2$.

cat()

Renvoie le contenu d'un fichier sous la forme d'une chaîne de caractères.

Format

cat(*filename*)

Paramètre

Paramètre	Définition
filename	Nom du fichier dont le contenu doit être renvoyé

Description

La fonction cat() renvoie le contenu du fichier *filename* sous la forme d'une seule chaîne, ou d'une chaîne NULL en cas d'erreur. Les retours à la ligne sont préservés afin de permettre l'utilisation de l'instruction foreach pour traiter la chaîne renvoyée comme une liste des lignes dans *filename*.

Exemple

Les instructions SMSL suivantes donnent les noms des utilisateurs inscrits dans le fichier de mots de passe du système UNIX.

```
people = cat("/etc/passwd");
foreach person (people)
{
name = ntharg(person, 1, ":");
printf("name of person is:%s", name, "\n");
}
```

ceil()

Renvoie le plus petit entier non inférieur à l'argument.

Format

ceil(*argument*)

Paramètre

Paramètre	Définition
argument	Argument numérique dont l'arrondi à l'entier supérieur le plus proche est à déterminer

Description

La fonction ceil() renvoie l'entier le plus petit non inférieur à *l'argument*; c'est-à-dire l'arrondi à l'entier supérieur le plus proche pour *argument*.

Les fonctions ceil() et floor() encadrent *l'argument* de la manière suivante:

- Si *argument* est un entier: ceil(*argument*) = *argument* = floor(*argument*).
- Si *argument* n'est pas un entier: floor(*argument*) < *argument* < ceil(*argument*) et ceil(*argument*) = floor(*argument*) + 1.

chan_exists()

Vérifie l'existence d'un processus ou d'un canal de fichier.

Format

chan_exists(*channel*)

Paramètre

Paramètre	Définition
channel	Nom (canal partagé) ou numéro (canal local) du canal E/S du processus ou du fichier à vérifier

Description

La fonction chan_exists() renvoie 1 si le canal local ou partagé existe et renvoie 0 s'il n'existe pas.

La valeur de retour de la fonction chan_exists() peut être utilisée avec des variables de condition pour synchroniser un processus SMSL afin d'attendre qu'un autre ait ouvert un canal en utilisant la fonction popen() ou la fonction fopen().

close()

Ferme un canal de fichier ou de processus.

Format

close(*channel*,*flags*)

Paramètres

Paramètre	Définition
channel	Nom (canal partagé) ou numéro (canal local) du canal E/S du processus ou du fichier à fermer
flags	Bits indicateurs optionnels utilisés pour contrôler l'exécution de fermeture. Les bits suivants sont utilisés: Bit 1 = 1 indique que n'importe quel processus de système associé au canal (c'est-à-dire la fonction SMSL fopen() ou popen()) doit être détruit lors de la fermeture du canal. Bit 2 = 1 indique qu'un canal doit être fermé même si un autre processus SMSL est bloqué en attendant une fonction read(), readln(), ou write(). Le bit 2 est valide uniquement pour les canaux globaux et non pour les canaux locaux. Par défaut si non spécifié: Les bits 1 et 2 sont à zéro

Description

La fonction close() ferme un canal à un processus ou à une commande préalablement créé par un appel fopen() ou popen().

Lorsque les indicateurs ne sont pas spécifiés, leur valeur par défaut est nulle.

Lorsque le bit 1 = 0, la fonction close() ne détruit pas les processus générés de manière dynamique par fopen() ou popen(); et ces processus sont autorisés à continuer. Cette caractéristique de close() permet d'ouvrir un canal à un processus SMSL, d'envoyer des données supplémentaires et de fermer le canal tout en permettant au processus de s'achever.

Lorsque le bit 2 = 1, la fonction close() fermera le canal même si un autre processus SMSL est bloqué en attente d'une demande E/S sur le canal en question. Lorsque le blocage a lieu, la fonction close() force la fonction bloquée à s'activer et à recevoir du processus par lequel le canal a été ouvert un retour d'erreur et un numéro errno.

La fonction close() renvoie une chaîne vide NULL si la fermeture a réussi et –1 avec la variable SMSL errno valuée si la fermeture n'a pas réussi. La fonction close() échoue lorsque le bit 2 = 0 et que le canal est un canal global avec au moins un processus SMSL bloqué.

Exemple

fermer le canal représenté par la variable chan

close(chan);

concat()

Concatène deux chaînes.

Format

concat(*string1*, *string2*)

Description

La fonction concat concatène les deux chaînes.

Par exemple, concat("ab","cd") renvoie "abcd".

cond_signal()

Signale un processus qui est bloqué en état d'attente sur condition.

Format

cond_signal(*condition_variable,all*)

Paramètres

Paramètre	Définition
condition_variable	Nom de la variable qui débloquera un processus bloqué par la fonction cond_wait()
all	Valeur non vide qui indique à la fonction cond_signal() de débloquer tous les processus SMSL bloqués en attente de condition_variable

Description

La fonction cond_signal() peut signaler un autre processus SMSL qui est bloqué pour une fonction cond_wait() sur *condition_variable*. Si *all* est spécifié et n'est pas dans la chaîne NULL, la fonction cond_signal() activera tous les processus SMSL qui sont bloqués sur condition_variable. Si aucun processus n'est bloqué sur condition_variable, la fonction cond_signal() n'a pas d'effet. La fonction cond_signal() ne peut jamais bloquer et renvoie toujours la chaîne NULL.

cond_wait()

Bloquer un processus jusqu'à réception d'un signal d'état.

Format

cond_wait(*condition_variable,lockname,timeout*)

Paramètres

Paramètre	Définition
condition_variable	Nom de la variable qui terminera l'état cond_wait(). *condition_variable* est émis par la fonction cond_signal().
lockname	Nom du verrou que la fonction cond_wait() doit tenter d'acquérir lorsqu'elle reçoit la *condition_variable* correcte dans la fonction cond_signal(). Si *lockname* est une chaîne NULL, la fonction cond_wait() ne tentera pas d'acquérir un verrou après avoir reçu la *condition_variable*.
timeout	Nombre de secondes d'attente pour la réception de *condition_variable* avant de débloquer et libérer le *lockname*. Plage valide: *timeout* > 0 spécifie la valeur de temporisation en secondes; *timeout* < 0 spécifie une temporisation infinie, seule la réception de *condition_variable* peut débloquer le processus; *timeout* = 0 n'est pas autorisée et donnera un message SMSL d'erreur à l'exécution comme résultat. Par défaut si non spécifié: Temporisation infinie.

Description

La fonction cond_wait() bloque le processus SMSL en cours jusqu'à réception de *condition_variable* ou expiration de la temporisation. Si le processus SMSL maintient *lockname* à l'émission de la fonction cond_wait(), la fonction cond_wait() libère le verrou. Lorsque la fonction cond_wait() reçoit condition_variable, la fonction cond_wait() tente immédiatement d'obtenir *lockname*.

Si la fonction cond_wait() renvoie 1, elle maintiendra toujours un verrou exclusif sur *lockname*. Si la fonction cond_wait() échoue, elle ne maintiendra aucun verrou sur *lockname* lorsqu'elle renverra. Si une temporisation a lieu, la fonction cond_wait() renvoie une valeur d'échec de "0", met le errno SMSL à E_SMSL_TIMEOUT et ne maintiendra aucun verrou sur *lockname*.

condition_variable

condition_variable est le nom de la variable d'état que la fonction cond_wait() attend et qui a été signalée par la fonction cond_signal(). Les noms des variables d'état ont une portée globale analogue aux verrous et aux canaux partagés.

Aucune de ces portées globales ne perturbe les autres. On peut utiliser le même nom sans interférer avec un verrou, un canal partagé ou une variable d'état.

lockname

En entrant dans la fonction cond_wait(), le processus libère le verrou *lockname* et les blocs en attente de signalisation. *lockname* est généralement un verrou exclusif maintenu par ce processus; sinon, des messages d'erreur à l'exécution risquent d'être générés (mais la fonction cond_wait() tentera quand même de continuer et d'attendre un signal). La fonction cond_wait() bloquera toujours l'attente de la fonction cond_signal() ou d'une temporisation.

Lorsqu'un autre processus SMSL exécute une fonction cond_signal() qui active ce processus SMSL, l'appel de la fonction cond_wait() tentera d'obtenir un verrou exclusif (si un verrou est demandé, c'est-à-dire si *lockname* n'est pas la chaîne vide) et renverra immédiatement le verrou ou rejoindra la file en attente d'un verrou exclusif sur *lockname*.

La pratique courante est de fournir le *lockname* puisque les variables d'état sont presque toujours bloquées par verrouillage. Dans la fonction cond_wait(), *lockname* doit être fourni comme la chaîne NULL plutôt qu'omis afin de permettre au codeur SMSL de décider si un verrou est nécessaire. Le *lockname* exigé réduira le nombre d'erreurs occasionnées par la non-utilisation d'un verrou qui était nécessaire.

timeout

Le comportement de la temporisation ne change pas, même si la fonction cond_wait() attend une fonction cond_signal() ou l'acquisition de *lockname*. Si *condition_variable* ou *lockname* sont arrêtés avant la fin de la fonction cond_wait(), la fonction cond_wait() renvoie 0 et établit la valeur de errno SMSL mais ne maintiendra pas de verrou sur *lockname*.

lockname peut être une chaîne NULL, dans ce cas, *condition_variable* est considérée de ne pas avoir de verrou associé et la fonction cond_wait() renverra immédiatement une valeur de réussite sans attendre un verrou.

cos()

Renvoyer le cosinus de l'argument.

Format

cos(*radians*)

Paramètre

Paramètre	Définition
radians	Longueur d'arc en radians dont le cosinus est à déterminer Plage valide: $-\infty \leq$ radians $\leq \infty$

Description

La fonction cos() renvoie le cosinus des radians.

La plage de résultat pour la fonction cos() est $-1 \leq \cos() \leq 1$.

cosh()

Renvoyer le cosinus hyperbolique de l'argument.

Format

cosh(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur numérique dont le cosinus hyperbolique est à déterminer Plage valide: $-\infty \leq$ *argument* $\leq \infty$

Description

La fonction cosh() renvoie le cosinus hyperbolique de l'argument. Le cosinus hyperbolique est défini par l'expression:

$$\cosh(x) = (e\ x + e\ –x\ )/2$$

où e est la base pour des logarithmes naturels (e = 2,71828 . . .). La plage de résultat pour la fonction cosh() est $1 \leq \cosh() \leq \infty$.

date()

Renvoyer la date et l'heure comme une chaîne de 24 caractères.

Format

date()

Description

La fonction date() renvoie la date et l'heure actuelles dans une chaîne de 24 caractères sous le format:

Sun Sep 16 01:03:52 1973

La fonction date() équivaut à la fonction ctime(3) de la bibliothèque C. La fonction date() équivaut également à l'instruction SMSL:

asctime(time());

Exemple

Les exemples suivants soulignent l'utilisation de la fonction date().

Affecter la date et l'heure courantes à une variable:

today = date();

debugger()

Suspendre le processus en attente d'un niveau de commande du débogueur SMSL.

Format

debugger()

Description

La fonction debugger() SMSL suspend le processus SMSL en cours en attente d'un niveau de commande du débogueur SMSL. La fonction debugger() complète les options à l'intérieur du débogueur SMSL qui suspendent les processus SMSL. La fonction debugger() offre une méthode de hiérarchie inférieure d'arrêt d'un processus SMSL pour le déboguer avant qu'il ne soit lancé.

Bien que la modification d'un code source SMSL pour déboguer un script particulier peut s'avérer inappropriée, la fonction debugger() fournit une méthode générale qui permet de déboguer tous les codes SMSL.

Le seul moyen de relancer une fonction SMSL suspendue par la fonction debugger() est d'utiliser le débogueur SMSL. Si le processus SMSL est déjà en cours de traitement par le débogueur SMSL, un appel lancé à la fonction debugger() n'a aucun effet. La fonction debugger() renvoie toujours une chaîne NULL.

destroy()

Détruire un objet SMSL.

Format

destroy(*object, description*)

Paramètres

Paramètre	Définition
object	Identificateur alphanumérique pour l'objet. L'objet est attribué quand l'objet est créé.
description	Chaîne de texte optionnelle qui peut être utilisée pour expliquer pourquoi l'objet a été détruit. La chaîne de texte doit être entre guillemets.

Description

La fonction destroy() supprime l'objet d'instance d'application. La fonction destroy() renvoie VRAI pour une réussite et FAUX pour une erreur.

Exemple

destroy object whose name is in variable <name>

destroy(name);

Par défaut si non spécifié: chaîne NULL

difference()

Renvoyer la liste d'éléments qui sont uniques pour une liste SMSL spécifiée.

Format

différence(*list1,list2,list3,list4 . . . ,listn*)

Paramètres

Paramètre	Définition
list1	Liste SMSL dont les éléments sont comparés aux éléments de toutes les autres listes spécifiées
list2 . . . listn	Une ou plusieurs listes en option dont les éléments sont comparés à *list1*

Description

La fonction différence() renvoie une liste SMSL avec tous les éléments de *list1* qui ne sont pas inclus dans les listes *list2 . . . listn*. Si *list1* est la liste NULL, le résultat est dans la liste NULL.

list1 peut contenir des doubles. Les doubles dans list1 apparaissent dans la liste retour dans le même ordre et quantité que dans *list1*, à condition qu'ils n'aient pas été retirés par des correspondances avec d'autres listes dans la fonction difference().

Tous les éléments renvoyés de *list1* conservent le même ordre dans la liste retour. Si la liste de retour n'est pas l'ensemble NULL, l'ensemble renvoyé est délimité par des caractères interligne; c'est-à-dire que tous les éléments d'ensemble se terminent par un caractère d'interligne.

execute()

Exécuter une commande d'un type spécifié.

Format

execute(*type,command,instance*)

Paramètres

Paramètre	Définition
type	Processeur de commandes qui doit interpréter et exécuter la commande Plage valide: Les types de commandes intégrées OS ou SMSL ou un type de commande valide défini par l'utilisateur.
command	Syntaxe de la commande concernée
instance	Instance d'application pour laquelle la commande doit exécuter Par défaut si non spécifié: L'instance d'application hiérarchiquement la plus proche de la commande.

Description

La fonction execute() exécute une commande de n'importe quel type et renvoie tout résultat qu'elle produit à stdout ou à stderr. L'état de la commande est sauvegardé dans la variable SMSL exit_status.

Exemple

```
# SQL data is returned into the buffer "data"

data = execute("SQL", "select * from user_objects");
```

exists()

Vérifier l'existence d'un objet SMSL.

Format

exists(*object,inherit*)

Paramètres

Paramètre	Définition
object	Identificateur alphanumérique pour l'objet dont on vérifie l'existence. *Object* est attribué quand l'objet est créé.
inherit	L'expression booléenne contrôle si exists recherchera dans toute la hiérarchie d'héritage afin de vérifier l'existence de *object*: Si *inherit* = VRAI, rechercher la hiérarchie d'héritage. Si *inherit* = FAUX et si l'objet n'est pas une référence à un objet absolu, rechercher la hiérarchie d'héritage.

Description

La fonction exists() renvoie VRAI si un objet existe; sinon, FAUX. La fonction exists() est utile dans des procédures de découverte d'application qui déterminent si une instance a déjà été découverte et instanciée dans la hiérarchie de l'objet.

Exemple

```
# Check if we have created the user before

if (exists(name))

{

        printf("%f",name);

}

else

{

        printf("User name does not exist");

}
```

exp()

Renvoyer la base des logarithmes naturels e élevée à une puissance.

Format

exp(*exponent*)

Paramètre

Paramètre	Définition
exponent	Valeur numérique à laquelle on élève la base naturelle e

Description

La fonction exp() renvoie la valeur $e^{exponent}$ où *e* est la base des logarithmes naturels (*e* = 2,71828 . . .).

fabs()

Renvoyer la valeur absolue d'un argument.

Format

fabs(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur de virgule flottante dont la valeur absolue est à déterminer

Description

La fonction fabs() renvoie la valeur absolue de l'argument, c'est-à-dire:

- *argument if argument* ≥ 0;
- *–argument if argument* < 0.

file()

Renvoyer l'information de fichier.

Format

file(*filename*,*dummy*)

Paramètres

Paramètre	Définition
filename	Nom du fichier dont la dernière date de modification est à renvoyer
dummy	Variable fictive qui spécifie l'information de fichier étendu sous la forme: modtime atime ctime mode size numlinks type modtime est la dernière date de modification exprimée en nombre de secondes depuis le 1er janvier 1970 à minuit. atime est la dernière heure d'accès exprimée en nombre de secondes depuis le 1er janvier 1970 à minuit. ctime est le dernier changement d'état exprimé en nombre de secondes depuis le 1er janvier 1970 à minuit. mode est l'autorisation de fichier exprimée en entier octal. size est la longueur du fichier exprimée en nombre de caractères. numlinks est le nombre de liaisons au fichier à l'intérieur du système de fichier. type est une chaîne de caractères indiquant le type de fichier: FILE fichier de données d'utilisateurs ordinaires DIR annuaire SPECIAL fichier spécial de caractères BLOCK fichier spécial de blocs FIFO canal ou FIFO LINK liaison symbolique SOCKET prise (pas disponible sur toutes les plates-formes) UNKNOWN type de fichier inconnu, peut-être un LINK ou SOCKET sur des plates-formes où la fonction UNIX stat() ne peut déterminer le type, c'est-à-dire lorsque S_ISLINK ou S_ISSOCK ne sont pas définis.

Description

La fonction file() renvoie la dernière heure de modification du fichier filename en nombre de secondes depuis le 1er janvier 1970 à minuit. Si le fichier n'existe pas, la fonction file() renvoie la chaîne NULL. Cette fonction est utile pour vérifier l'existence d'un fichier.

NOTE 1 – La fonction file() renvoie des valeurs par rapport au système d'exploitation et, dans certains cas, par rapport au système de fichiers. Certaines plates-formes non-UNIX ne renvoient pas toutes les valeurs de retour ou peuvent renvoyer une ou plusieurs valeurs de retour sans signification. Pour une plate-forme spécifique, la fonction file() renverra généralement les mêmes informations que la fonction stat() du langage de programmation C.

L'utilisateur n'a pas besoin d'autorisation pour lire le fichier mais en a besoin pour rechercher chaque répertoire dans le chemin d'accès qui mène au *filename*. Si l'utilisateur ne dispose pas d'une telle autorisation, la fonction file() échoue et renvoie la chaîne NULL.

On ne tient pas compte de la valeur de *dummy*, mais sa présence force la fonction file() à renvoyer une chaîne d'informations plus détaillée.

Exemples

Les exemples suivants soulignent l'utilisation de la fonction file().

Print Last Modification Date of the UNIX System Password File

```
printf("%s",asctime(file("/etc/passwd")));
# modification time
```

Test for the Existence of a File

```
if (file("some_file"))
{
    printf("File exits!");
}
else
{
    printf("File does not exist.");
}
```

floor()

Renvoyer le plus grand entier qui n'est pas supérieur à l'argument.

Format

floor(*argument*)

Paramètre

Paramètre	Définition
argument	Argument numérique dont l'arrondi à l'entier inférieur le plus proche est à déterminer

Description

La fonction floor() renvoie le plus grand entier qui n'est pas supérieur à *argument*; c'est-à-dire l'entier inférieur le plus proche de *argument*.

La fonction floor() et la fonction ceil() encadrent *argument* de telle manière que:

Si *argument* est un entier: ceil(*argument*) = *argument* = floor(*argument*)

Si *argument* n'est pas un entier: floor(*argument*) < *argument* < ceil(*argument*) et ceil(*argument*) = floor(*argument*) + 1

fmod()

Renvoyer le reste de la division en virgule flottante.

Format

fmod(*dividend*,*divisor*)

Paramètres

Paramètre	Définition
dividend	Valeur de virgule flottante dont le reste sera renvoyé après avoir été divisé par *divisor*
divisor	Valeur de virgule flottante qui divisera *dividend*

Description

La fonction fmod() renvoie le reste de la division en virgule flottante (*dividend*)/(*divisor*).

fopen()

Ouvrir un canal SMSL à un fichier.

Format

fopen(*filename*,*mode*)

Paramètres

Paramètre	Définition
filename	Nom du fichier auquel le canal SMSL doit être ouvert
mode	Mode d'accès au fichier. Plages valides: r Ouvrir pour lecture w Tronquer à zéro ou créer un fichier pour écrire a Ouvrir pour ajouter à la fin du fichier ou créer pour écrire rb Ouvrir fichier binaire pour lecture wb Tronquer fichier binaire à zéro ou créer fichier binaire pour écrire ab Ouvrir fichier binaire pour ajouter à la fin du fichier ou créer fichier binaire pour écrire r+ Ouvrir pour lecture et écriture (*update*) w+ Tronquer à zéro ou créer pour lire et écrire a+ Ouvrir pour lire et écrire à la fin du fichier ou créer fichier pour lire et écrire r+b Ouvrir fichier binaire pour lire et écrire (*update*) w+b Tronquer fichier binaire à zéro ou créer fichier binaire pour lire et écrire a+b Ouvrir fichier binaire pour lire et écrire à la fin du fichier ou créer fichier binaire pour lire et écrire

Description

La fonction fopen() ouvre un canal à filename qui fournit l'accès à filename à l'intérieur d'un processus SMSL. Les fonctions read(), write(), get_chan_info(), share(), et close() s'appliquent à des canaux qui ont été ouverts à des fichiers.

Lorsqu'elle est prise en charge par le système d'exploitation sous-jacent, la fonction fopen() exécute des contrôles de sécurité afin de déterminer si le nom de l'utilisateur du processus d'appel a l'autorisation pour la demande.

Si la fonction fopen() réussit, elle renvoie le numéro de canal SMSL au filename. Un échec d'ouverture de filename, tel qu'un problème au niveau du système d'exploitation ou un mode non valide, sélectionne la valeur de errno SMSL et force la fonction fopen() à renvoyer la chaîne NULL sans tenter d'ouvrir le fichier.

Prise en charge pour accès au fichier binaire

La fonction fopen() autorise des modes binaires avec un caractère b bien qu'il n'y ait pas de moyen d'écrire, à l'intérieur d'un processus SMSL, des formes de données binaires autres que des chaînes de caractères.

Vider un fichier après chaque opération du fichier SMSL

Les fonctions SMSL s'assurent qu'un fichier est vidé après chaque opération afin d'éviter de commettre dans les opérations de fichier SMSL l'erreur courante d'exécuter un write-then-read ou un read-then-write sans l'intermédiaire d'un fseek rewind ou d'un fflush.

fseek()

Mettre l'indicateur de position de fichier.

Format

fseek(*channel,offset,whence*)

Paramètres

Paramètre	Définition
channel	Canal de fichier E/S renvoyé quand le fichier a été ouvert par la fonction fopen()
offset	Nombre d'octets à ajouter à whence pour obtenir la position de fichier
whence	Point standard dans un fichier auquel on ajoute offset pour obtenir la nouvelle position du fichier Plage valide: Une des valeurs d'entier suivantes: 0 SEEK_SET, le début du fichier 1 SEEK_CURR, la position de fichier actuelle 2 SEEK_END, la fin du fichier

Description

La fonction fseek() met l'indicateur de position filename à la position whence plus des octets offset. Si whence n'est pas valide, la fonction fseek() prend implicitement la valeur whence = 0 et applique une erreur à l'exécution mais termine l'opération de recherche de fichier.

NOTE 2 – L'application de la fonction fseek() avec des fichiers binaires où whence = 2 (SEEK_END) n'est pas réellement prise en charge sur toutes les plates-formes.

La fonction fseek() renvoie 0 pour une réussite et –1 pour un échec. Pour un canal non valide, c'est-à-dire pour un canal de communication au lieu d'un canal de fichier, la fonction fseek() renvoie –1, applique une erreur à l'exécution et sélectionne la variable errno SMSL.

fseek et mode append file

L'utilisation de la fonction fseek() pour changer l'indicateur de position du fichier dans un fichier ouvert en mode ajout, c'est-à-dire les modes a, ab ou a+, n'empêchera pas les écritures à la fin du fichier en utilisant la fonction write().

Exemple

Le SMSL ne contient aucun équivalent à la fonction C rewind(), mais l'exemple suivant de la fonction fseek() équivaut à la fonction C rewind(*channel*):

```
fseek(channel,0,0);
```

ftell()

Renvoyer l'indicateur de position de fichier.

Format

ftell(*channel*)

Paramètre

Paramètre	Définition
channel	Canal de fichier E/S renvoyé quand le fichier a été ouvert par la fonction fopen()

Description

La fonction ftell() renvoie l'indicateur de position de fichier comme l'entier d'octets du début du fichier. Pour un canal non valide, c'est-à-dire un canal de communication au lieu d'un canal de fichier, la fonction ftell() renvoie –1, applique une erreur à l'exécution et sélectionne la variable errno SMSL.

Le résultat type des versions C et SMSL de la fonction ftell() est le nombre de caractères écrits ou lus dans un fichier, sauf sur les plates-formes qui exécutent des conversions d'interligne CR/LF sur des fichiers de texte. Cependant, la valeur de la fonction ftell() après avoir exécuté une fonction fseek() à la fin du fichier donne généralement le nombre total de caractères dans le fichier.

Les fonctions SMSL suivantes changent l'indicateur de position de fichier:

- fopen();
- fseek();
- read();
- readln();
- write().

La fonction get_chan_info() ne change pas l'indicateur de position de fichier. La fonction close() rend *channel* non valide.

full_discovery()

Vérifier que le processus est actuellement dans un cycle full discovery.

Format

full_discovery()

Description

La fonction full_discovery() renvoie VRAI si le script SMSL qui le contient est un script de découverte d'application et s'il est actuellement dans un cycle full discovery. Sinon, la fonction full_discovery() renvoie FAUX.

Un cycle full discovery est exécuté après le rafraîchissement de l'antémémoire du processus de l'agent. Cet indicateur signale, par conséquent, si l'antémémoire du processus a été rafraîchi depuis la dernière exécution du script.

Exemple

L'exemple suivant vérifie si le script SMSL est en cycle full discovery et sinon, il quitte le script.

```
# If we are not in a full discovery cycle
# we can exit immediately
if (!full_discovery())
{
    exit;
}
```

get()

Renvoyer la valeur actuelle d'une variable.

Format

get(*variable*)

Paramètre

Paramètre	Définition
variable	Nom de la variable dont la valeur actuelle sera renvoyée

Description

La fonction get() renvoie la valeur actuelle de la variable. Si *variable* est un nom relatif et n'existe pas dans le contexte du script SMSL, la fonction get() recherche successivement le contexte hiérarchique jusqu'à ce qu'elle trouve *variable* ou jusqu'à ce que la recherche échoue dans le contexte de l'ordinateur.

Exemple

L'exemple suivant renvoie l'état actuel de la base de données RDB Dev.

```
get ("/RDB/Dev/status");
```

get_chan_info()

Renvoyer les informations d'état d'un fichier SMSL ou d'un canal de processus.

Format

get_chan_info(*channel*,*flags*)

Paramètres

Paramètre	Définition
channel	Nom de canal (canaux partagés) ou nombre (canaux locaux) dont l'état doit être signalé ou "" indiquant que tous les canaux sont à signaler (soumis au contrôle d'indicateurs)
flags	Valeur de l'entier représentant deux indicateurs binaires qui contrôle la sortie des informations de canaux globaux et locaux comme suit: 1 uniquement un canal global 2 uniquement deux canaux locaux 3 trois canaux, globaux et locaux Par défaut si non spécifié: 1

Description

La fonction get_chan_info() renvoie des informations de canal, une chaîne avec le format

name status details type scope read_pid read_name write_pid write_name

spécifiant:

```
get_chan_info("");
```

force la fonction get_chan_info() à renvoyer des descriptions pour tous les canaux globaux partagés. Les descriptions sont structurées en une liste séparée par des interlignes, une ligne par canal.

Définition de champ

name Parmi les suivants:

- scope=SHARED – nom de canal.
- scope=LOCAL – nombre de canaux locaux.
- scope="" – tous les canaux partagés et locaux.

status OUVERT ou FERMÉ

details Parmi les suivants:

- fopen() channel – nom du fichier ouvert ou AUCUN si aucun nom de fichier n'est ouvert;
- popen() channel – ID de processus du processus du système d'exploitation externe auquel le canal est associé; ou
- –1 si le processus est terminé.

type CANAL ou FICHIER

scope PARTAGÉ ou LOCAL

read_pid Parmi les suivants:

- ID de processus du processus SMSL en attente de lecture du canal.
- –1 si aucun processus n'est en attente.

read_name Parmi les suivants:

- Nom du processus en attente de lecture du canal.
- AUCUN si aucun processus n'est en attente.
- NON DISPONIBLE s'il existe un processus mais que le nom n'est pas disponible.

write_pid Parmi les suivants:

- ID de processus en attente d'écrire vers le canal.
- –1 si aucun processus n'est en attente.

write_name Parmi les suivants:

- Nom du processus en attente d'écrire vers le canal.
- AUCUN si aucun processus n'est en attente.
- NON DISPONIBLE s'il existe un processus mais que le nom n'est pas disponible.

Spécifiant

get_chan_info("",flags);

force la fonction get_chan_info() à renvoyer tous les canaux locaux ou globaux pour le processus SMSL actuel tels que contrôlés par la valeur des indicateurs. Noter que les fonctions get_chan_info() suivantes sont équivalentes, pour les deux il faut renvoyer la liste des canaux globaux:

get_chan_info("");

get_chan_info("",1);

La fonction get_chan_info() génère un avertissement à l'exécution si les indicateurs sont non numériques ou ne sont pas supérieurs à zéro, mais la variable errno SMSL n'est mise à aucune valeur. L'interpréteur SMSL ne tient pas compte des indicateurs sans erreur si le canal n'est pas la chaîne vide.

La fonction get_chan_info() renvoie tous les champs pour chaque canal même s'ils ne s'appliquent pas à un canal particulier.

La fonction get_chan_info() renvoie la chaîne NULL si:

- il n'y a aucun canal partagé global et/ou canal local pour la valeur donnée des indicateurs;
- elle reçoit un numéro de canal incorrect ou un mauvais nom.

Dans ce cas, la fonction get_chan_info() sélectionne également la variable errno SMSL.

getenv()

Renvoyer la valeur de chaîne d'une variable d'environnement SMSL.

Format

getenv(*variable*)

Paramètre

Paramètre	Définition
variable	Nom de l'objet dont la valeur est à renvoyer

Description

La fonction getenv() renvoie la valeur de chaîne d'une variable dans l'environnement du script SMSL. La valeur de variable peut être renvoyée à partir des endroits suivants:

- les variables d'environnement définies pour le paramètre;
- les variables d'environnement définies pour l'application;
- l'environnement défini pour l'ordinateur;
- l'environnement de l'agent au début de son exécution.

La fonction getenv() recherche les tables d'environnement dans l'ordre établi et renvoie la valeur de la première variable correspondante. La fonction getenv() renvoie la chaîne NULL si la variable n'est pas définie et met la variable errno SMSL à une valeur non zéro. Si la fonction getenv() réussit, elle renvoie la valeur de la variable et met la variable errno SMSL à zéro. On peut donc utiliser errno pour différencier une variable non définie d'une variable qui est mise à la chaîne NULL.

Exemple

Cet exemple SMSL présente une fonction qui vérifie si une variable d'environnement existe.

```
function is_environment_var_defined(name)
{
    getenv(name); # Throw away return value of getenv
    return (errno == 0); # errno is only zero if name is defined
}
```

get_vars()

Renvoyer la liste de variables pour un objet SMSL.

Format

get_vars(*object,showchildren*)

Paramètres

Paramètre	Définition
object	Nom optionnel de l'objet dont les variables sont à lister Par défaut si non spécifié: Objet actuel
showchildren	Indicateur optionnel dont la valeur non zéro indique que get_vars() doit également inscrire les sous-objets de l'objet. Par défaut si non spécifié: 0

Description

La fonction get_vars() renvoie une liste des variables d'objet ou de l'objet actuel si l'objet est omis. La fonction get_vars() renvoie la chaîne NULL si l'objet n'existe pas.

La liste de variables d'objet est triée en ordre alphabétique croissant.

grep()

Renvoyer les lignes d'un bloc de texte qui correspondent à une expression normale.

Format

grep(*regular_expression,text,v*)

Paramètres

Paramètre	Définition
regular_expression	Séquence de caractères qui définit la configuration que la fonction grep recherche dans le texte. *regular_expression* est conforme aux expressions normales définies dans la commande UNIX ed(1) et dans la description UNIX regexp(5). Un résumé des différents caractères d'expression normale est donné ci-dessous: ^beginning of line \< beginning of a word $ end of line \> end of a word. match any single character * match zero or more repetitions of the preceding [] match any of the characters contained within [^] match any characters except those contained within.
text	Texte à rechercher pour correspondre à regular_expression. text peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
v	Le caractère v inverse le résultat de la fonction grep() forçant la fonction grep() à sortir toutes les lignes en texte qui ne contiennent pas une correspondance pour regular_expression. Cet indicateur est similaire à celui de UNIX grep -v.

Description

La fonction grep() renvoie une liste de lignes en texte qui correspond à *regular_expression*.

Si un caractère autre que v est transmis comme indicateur d'inversion de la fonction grep(), l'interpréteur SMSL renvoie un message d'erreur à l'exécution.

Exemple

```
# search for "martin" substring in /etc/passwd

all_lines=cat("/etc/passwd");

# fill a buffer with passwd

matching_lines=grep("martin",all_lines);
```

history()

Renvoyer l'information history de la base de données history.

Format

history(*parameter*,*format*,*number*)

Paramètres

Paramètre	Définition
parameter	Nom de l'objet dont l'historique doit être renvoyé. L'expression "" indique le paramètre actuel. Le paramètre peut être: the absolute path, such as "/APP/INST/PARAM" a relative path, such as "." or "../DIFFERENTINST/PARAM". Le paramètre peut être "" ou "." pour l'historique du paramètre actuel. Par défaut si non spécifié: Paramètre actuel
format	Chaîne de caractères optionnelle entre guillemets qui spécifie le format de chaque entrée de la fonction history(). Valeurs valides: n renvoyer le nombre de points de données disponibles comme la première valeur dans la liste de retour t, inclure l'heure et la date de chaque entrée dans la liste de retour v et inclure la valeur de chaque entrée historique dans la liste de retour. Par défaut si non spécifié: ntv
number	Valeur numérique optionnelle qui limite le nombre d'entrées que la fonction history renverra. Par défaut si non spécifié: 50

Description

La fonction history() a accès au paramètre de la base de données historiques et renvoie une liste contenant le nombre de points de données disponibles suivie par le nombre d'entrées.

La fonction history() renvoie la chaîne vide, génère une erreur à l'exécution et sélectionne la variable errno SMSL si un mauvais caractère de format est fourni.

Du fait des valeurs par défaut fournies dans la fonction history(), les spécifications de fonction suivantes sont équivalentes:

```
history(parameter)

history(parameter,"ntv",50)
```

Format de résultat historique

La fonction history() renverra un des formats suivants en fonction de la sélection des indicateurs de format:

- *number_entries*\n si l'indicateur n est sélectionné;
- value\n, time\n, ou value time\n si les indicateurs v, t et vt sont sélectionnés.

La fonction history() sépare les valeurs d'une entrée par des espaces et des entrées successives par des caractères d'interligne.

On peut utiliser la fonction nthline(list1) pour obtenir le nombre de points de la tête de la liste ainsi que pour extraire les entrées. Les entrées peuvent être divisées si nécessaire en valeurs de temps et de données en utilisant la fonction ntharg(). Les entrées seront des valeurs simples si l'indicateur t ou v est absent.

index()

Renvoyer la position de départ d'une chaîne à l'intérieur d'une autre.

Format

index(*text,string*)

Paramètres

Paramètre	Définition
text	Texte à rechercher pour la présence d'une chaîne. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
string	Un ou plusieurs caractères entre guillemets à situer dans le texte.

Description

La fonction index() renvoie la position dans le texte à laquelle la chaîne commence ou 0 si une chaîne n'est pas présente dans le texte. La première position dans le texte est la position 1.

int()

Renvoyer le plus grand entier qui n'est pas supérieur à l'argument.

Format

int(*number*)

Paramètre

Paramètre	Définition
number	Valeur numérique ou variable numérique

Description

La fonction int() renvoie le plus grand entier qui n'est pas supérieur à *number*.

internal()

Traiter une commande interne à l'agent.

Format

internal(*command*)

Paramètre

Paramètre	Définition
command	Chaîne de texte qui est la commande que l'agent doit traiter.

Description

La fonction internal() force l'agent à traiter la commande de chaîne en interne et de manière spécifique à la plate-forme. La fonction internal() renvoie le résultat de commande si elle réussit. En cas d'échec ou si la commande n'est pas prise en charge sur la plate-forme spécifique, la fonction internal() renvoie la chaîne NULL et met la variable errno SMSL à E_SMSL_NOT_SUPPORTED.

La fonction internal() est destinée aux utilisateurs lors de la surveillance du processus et des demandes de ressources qui peuvent être traitées à l'intérieur de l'agent. Dans ces cas, il est plus efficace d'utiliser la fonction internal() plutôt que d'invoquer une commande séparée qui nécessite un appel à l'interpréteur SMSL ou à un autre processeur de commandes.

intersection()

Renvoyer une liste qui contient des éléments qui sont communs à toutes les listes spécifiées.

Format

intersection(*list1,list2,list3,list4 . . . listn*)

Paramètres

Paramètre	Définition
list1 . . . listn	Deux listes SMSL ou plus qui sont évaluées pour des éléments communs

Description

La fonction intersection() renvoie une liste SMSL qui contient les éléments qui apparaissent dans toutes les listes *list1. . . listn.*

La liste renvoyée n'est pas bien définie et contiendra des doubles s'ils étaient présents dans toutes les listes dans la même quantité et le même ordre. Les éléments de la liste renvoyée par la fonction intersection apparaissent dans le même ordre dans lequel ils figuraient dans list1.

Si une liste est une liste NULL, la valeur de retour est la liste NULL, sinon, toutes les entrées dans la liste renvoyée se terminent par un caractère d'interligne.

isnumber()

Vérifier qu'une chaîne est une représentation numérique valide.

Format

isnumber(*string*)

Paramètre

Paramètre	Définition
string	Chaîne qui doit être évaluée comme étant conforme aux critères d'une expression numérique

Description

La fonction isnumber() renvoie une valeur booléenne de 1 si la variable est une chaîne qui est considérée comme valide en tant que nombre et sinon, elle renvoie "0".

Un nombre valide est composé uniquement de chiffres, de points ou de signes moins pour chaque caractère dans la variable. Des espaces ou tout autre caractère non valide à n'importe quel endroit dans la chaîne forcent la fonction isnumber() à renvoyer 0. La fonction isnumber() renvoie 0 pour la chaîne NULL.

is_var()

Vérifier que la variable d'objet SMSL existe.

Format

is_var(*object*)

Paramètre

Paramètre	Définition
object	Nom de l'objet à vérifier comme une variable

Description

La fonction is_var() renvoie VRAI si l'objet existe et si c'est une variable. La fonction is_var() renvoie FAUX si:

- l'objet n'existe pas;
- l'objet existe mais n'est pas une variable (c'est-à-dire c'est une instance d'application ou un paramètre).

length()

Renvoyer le nombre de caractères dans une chaîne.

Format

length(*text*)

Paramètre

Paramètre	Définition
text	Texte dont on compte la longueur en caractères. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui donnent un texte en résultat.

Description

La fonction length() renvoie la longueur du texte en caractères y compris les interlignes.

lines()

Renvoyer le nombre de lignes dans une chaîne

Format

lines(*text*)

Paramètre

Paramètre	Définition
text	Texte dont on compte le nombre de lignes (c'est-à-dire des caractères d'interligne). text peut être le nom d'un fichier de texte, une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui donnent un texte en résultat.

Description

La fonction lines() renvoie le nombre de caractères d'interligne dans le texte. La fonction lines() est utile pour renvoyer la longueur d'une liste car les éléments d'une liste sont délimités par des interlignes.

lock()

Saisir un verrou de processus SMSL.

Format

lock(*lockname*,*mode*,*timeout*)

Paramètres

Paramètre	Définition
lockname	Nom du verrou à saisir
mode	Contrôle optionnel permis sous verrouillage. Plage valide: s partagé r lecture w écriture x exclusif Par défaut si non spécifié: x Si la première lettre du mode n'est pas s, r, w, ou x, une erreur à l'exécution a lieu et le mode par défaut se met à x (exclusif).
timeout	Valeur d'entier optionnelle qui spécifie le nombre de secondes avant l'expiration de la demande de verrouillage. Plage valide: timeout > 0 est l'entier du nombre de secondes pour lequel la demande de verrouillage est valide timeout = 0 signifie demande de verrouillage sans blocage timeout < 0 signifie une période de temporisation infinie (c'est-à-dire attente jusqu'à ce que le verrou soit débloqué) Par défaut si non spécifié: Temporisation infinie

Description

La fonction lock() demande un verrou avec le nom *lockname*. Le mode de la demande spécifie l'accès partagé (*reader*) ou exclusif (*writer*) sous verrouillage. La temporisation optionnelle spécifie le nombre de secondes pendant lequel la demande est valide.

Le comportement par défaut de la fonction lock() demande un verrouillage exclusif avec une période de temporisation infinie. La fonction lock() renvoie 1 pour une réussite ou 0 pour un échec.

Verrouillage et SMSL

Les noms de verrou sont globaux à l'agent, ainsi:

- tous les processus SMSL partagent la même table de verrouillage;
- des processus SMSL différents peuvent partager des noms de verrouillage de paramètre afin d'exécuter des actions concurrentes.

Il n'existe aucun moyen d'exécuter la détection de nom de verrou. Il est recommandé que les noms des verrous dans les programmes SMSL soient codés spécifiquement en utilisant le nom de l'application. Cette pratique évitera des conflits potentiels avec d'autres programmes SMSL.

Demandes de verrouillage partagé

Les demandes de verrouillage partagé pour un verrou qui se trouve en mode partagé sont accordées – sauf si une demande d'écriture est en attente. Donner la priorité à une demande d'écriture en attente empêche le mécanisme de verrouillage de bloquer les processus d'écriture.

Demandes de verrouillages déjà obtenus

Il est possible de demander un verrouillage déjà obtenu et bien que ce ne soit pas conseillé:

- la demande d'un verrouillage déjà obtenu n'est pas prise en compte;
- la demande d'un verrouillage partagé pour un verrou déjà obtenu avec accès exclusif n'est également pas prise en compte.

La demande d'une autorisation supérieure d'accès en exclusif d'un verrou actuellement partagé réussit et fait passer le verrou en exclusif à condition qu'il s'agisse du seul processus qui utilise le verrouillage. Si plusieurs processus utilisent le verrou, la fonction lock() renvoie immédiatement 0 en mode sans blocage (quelle que soit la valeur de la temporisation car le blocage entraînerait l'interblocage immédiat en vous attendant).

Au lieu d'utiliser cette demande, il est recommandé d'appeler la fonction unlock() afin de débloquer le verrouillage partagé avant de tenter de saisir le nouveau verrouillage exclusif. Il est possible de tracer des verrouillages en utilisant la variable SMSLDebug. La variable SMSLDebug peut être utile pour le déboguage des interactions de verrouillage à multiples processus.

Echec de la fonction verrouillage

La fonction lock() peut échouer si:

- une demande sans blocage échoue;
- une temporisation est dépassée avant que le verrouillage ne soit alloué.

La fonction lock() peut échouer pour une temporisation infinie si:

- une demande d'évolution spéciale est allouée;
- le système a exécuté des corrections d'interblocage externe.

loge()

Renvoyer le logarithme naturel de l'argument.

Format

loge(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur numérique dont le logarithme naturel est à déterminer. Plage valide: argument > 0

Description

La fonction loge() renvoie le logarithme de l'argument par rapport au logarithme naturel base e = 2,71828. . . La plage de résultat pour la fonction loge() est $-\infty < \text{loge}() < \infty$.

log10()

Renvoyer le logarithme en base 10 de l'argument.

Format

log10(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur numérique dont le logarithme en base 10 est à déterminer. Plage valide: argument > 0

Description

La fonction log10() renvoie le logarithme de l'argument par rapport à la base 10.

La plage de résultat pour la fonction log10() est $-\infty < \text{log10}() < \infty$.

ntharg()

Renvoyer une liste structurée contenant les champs d'une chaîne de texte.

Format

ntharg(*text,arguments,delimiters,separator*)

Paramètres

Paramètre	Définition
text	Texte à séparer en champs par la fonction ntharg(). *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
arguments	Liste d'entiers spécifiant les numéros de champs que la fonction ntharg() doit rechercher dans chaque ligne du texte. Les champs sont spécifiés comme suit: *x,y* champ x et champ y; *x-y* tous les champs de x à y inclus; *-x* tous les champs de 1 à x inclus; *x-* tous les champs de x au caractère d'interligne inclus
delimiters	Un ou plusieurs caractères que la fonction ntharg() doit traiter comme des délimiteurs de champs lors de l'examen du texte. Par défaut si non spécifié: space et \t (*tab*)
separator	Caractère optionnel à placer entre chaque champ du résultat de la fonction ntharg() Par défaut si non spécifié: caractère d'interligne ()

Description

La fonction ntharg() renvoie les arguments en texte.

Habituellement, la fonction ntharg() interprète chaque ligne dans le texte comme étant une liste de champs séparée par des espaces (espaces ou tabulations). Si les délimiteurs sont fournis, elle spécifie la liste de caractères que la fonction ntharg() doit traiter comme des séparateurs de champ. La fonction ntharg() renvoie généralement des champs sélectionnés comme une liste délimitée par des interlignes. Si un séparateur est fourni, elle spécifie le délimiteur à placer entre les éléments dans la liste renvoyée.

NOTE 3 – La différence entre la fonction ntharg() et la fonction nthargf() est la suivante:

- La fonction ntharg() traite chaque délimiteur qui suit un caractère non délimiteur comme étant la fin du champ. La fonction ntharg() interprète deux délimiteurs adjacents ou plus comme un seul délimiteur.
- La fonction nthargf() traite chaque délimiteur comme la fin du champ. La fonction nthargf() interprète deux délimiteurs adjacents ou plus comme étant les délimiteurs d'une ou de plusieurs chaînes NULL dont le contenu peut être demandé et renvoyé.

Exemple

L'exemple suivant affiche le nom d'ouverture de session et l'annuaire personnel de chaque utilisateur inscrit dans le fichier de mots de passe du système UNIX.

```
foreach user (cat("/etc/passwd"))
{
    printf(ntharg(user,"1,6",":","\t"),"\n");
}
```

nthargf()

Renvoyer une chaîne structurée contenant des champs d'une chaîne de texte.

Format

nthargf(*text*,*arguments*,*delimiters*,*separator*)

Paramètres

Paramètre	Définition
text	Texte à séparer en champs par la fonction nthargf(). *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
arguments	Liste d'entiers spécifiant les numéros de champs que la fonction nthargf() doit rechercher dans chaque ligne du texte. Les champs sont spécifiés comme suit: *x*,*y* champ x et champ y; *x*-*y* tous les champs de x à y inclus; -*x* tous les champs de 1 à x inclus; *x*- tous les champs de x au caractère d'interligne inclus
delimiters	Un ou plusieurs caractères que la fonction nthargf() doit traiter comme des séparateurs de champs lors de l'examen du texte. La fonction nthargf() interprète chaque présence de délimiteurs comme la délimitation d'un champ. La fonction nthargf() interprète deux délimiteurs adjacents ou plus comme un ou plusieurs champs NULL. Par défaut si non spécifié: space et \t (*tab*)
separator	Caractère optionnel à placer entre chaque champ du résultat de la fonction nthargf() Par défaut si non spécifié: caractère d'interligne ()

Description

La fonction nthargf() renvoie les arguments en texte.

Habituellement, la fonction nthargf() interprète chaque ligne dans le texte comme une liste de champs séparée par des espaces (espaces ou tabulations). Si les délimiteurs sont fournis, elle spécifie la liste de caractères que la fonction nthargf() doit traiter comme des séparateurs de champs. La fonction nthargf() renvoie généralement des champs sélectionnés comme une liste délimitée par un interligne. Si un séparateur est fourni, elle spécifie le délimiteur à placer entre les éléments de la liste renvoyée.

NOTE 4 – La différence entre la fonction nthargf() et la fonction ntharg() est la suivante:

- La fonction nthargf() traite chaque délimiteur comme la fin d'un champ. La fonction nthargf() interprète deux délimiteurs adjacents ou plus comme la délimitation d'une ou plusieurs chaînes NULL dont le contenu peut être demandé et renvoyé.
- La fonction ntharg() traite chaque délimiteur qui suit un caractère non délimiteur comme la fin d'un champ. La fonction ntharg() interprète deux délimiteurs adjacents ou plus comme un seul délimiteur.

nthline()

Renvoyer des lignes spécifiées d'une chaîne de texte.

Format

nthline(*text,lines,separator*)

Paramètres

Paramètre	Définition
text	Texte à séparer en lignes par la fonction nthline(). *text* peut être une chaîne de texte entre guillemets, ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
lines	Liste d'entiers spécifiant les numéros de lignes que la fonction nthline() doit rechercher dans le texte. Les lignes sont spécifiées comme suit: *x,y* ligne x et ligne y; *x-y* toutes les lignes de x à y inclus; *-x* toutes les lignes de 1 à x inclus; *x-* toutes les lignes de x au caractère EOF inclus
separator	Caractère optionnel à placer entre chaque champ du résultat de la fonction nthline() Par défaut si non spécifié: caractère d'interligne ()

Description

La fonction nthline() renvoie les lignes du texte séparées par des caractères d'interligne. Si un séparateur est spécifié, la fonction nthline() utilisera le séparateur pour séparer les lignes.

NOTE 5 – La différence entre les fonctions nthlinef() et nthline() est la suivante:

- La fonction nthlinef() traite chaque caractère d'interligne comme une ligne.
- La fonction nthline() traite uniquement une ligne non vide (c'est-à-dire une ligne avec un caractère non d'interligne précédant un caractère d'interligne) comme une ligne.

Exemple

Le script SMSL suivant affiche les cinq premiers processus qui s'exécutent sur un système UNIX.

```
# print the top five processes
printf("%s", nthline(system("ps -eaf"),"2-6"));
```

nthlinef()

Renvoyer les lignes spécifiées d'une chaîne de texte.

Format

nthlinef(*text,lines,separator*)

Paramètres

Paramètre	Définition
text	Texte à séparer en lignes par la fonction nthlinef(). *text* peut être une chaîne de texte entre guillemets, ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
lines	Liste d'entiers spécifiant les numéros de lignes que la fonction nthlinef() doit rechercher dans le texte. Les lignes sont spécifiées comme suit: *x,y* ligne x et ligne y; *x-y* toutes les lignes de x à y inclus; *-x* toutes les lignes de 1 à x inclus; *x-* toutes les lignes de x au caractère EOF inclus
separator	Caractère optionnel à placer entre chaque champ du résultat de la fonction nthlinef() Par défaut si non spécifié: caractère d'interligne ()

Description

La fonction nthlinef() renvoie les lignes du texte séparées par des caractères d'interligne. Si un séparateur est spécifié, la fonction nthlinef() utilisera le séparateur pour séparer les lignes.

NOTE 6 – La différence entre les fonctions nthlinef() et nthline() est la suivante:

- La fonction nthlinef() traite chaque caractère d'interligne comme une ligne.
- La fonction nthline() traite seulement une ligne non vide (c'est-à-dire avec un caractère non d'interligne qui précède un caractère d'interligne) comme une ligne.

Il est recommandé d'utiliser la fonction nthlinef() afin d'être homogène avec les autres fonctions SMSL.

Exemple

Le script SMSL suivant affiche les cinq premiers processus qui s'exécutent sur un système UNIX.

```
# print the top five processes
printf("%s",nthlinef(system("ps -eaf"),"2-6"));
```

popen()

Ouvrir un canal SMSL à un processus.

Format

popen(*type,command,instance*)

Paramètres

Paramètre	Définition
type	Processeur de commandes qui doit interpréter et exécuter la commande Plage valide: Les types de commandes intégrées OS ou SMSL ou un type de commande valide défini par l'utilisateur
command	Syntaxe de la commande émise
instance	Instance d'application avec laquelle la commande doit exécuter Par défaut si non spécifié: Instance d'application hiérarchiquement la plus proche de la commande

Description

La fonction popen() génère de manière dynamique un processus afin d'exécuter une commande d'un type défini et renvoie un numéro de canal qui peut être utilisé pour lire les résultats de la commande ou pour écrire des messages à la commande.

La fonction popen() renvoie –1 si erreur.

pow()

Porter un numéro à une puissance.

Format

pow(*base,exponent*)

Paramètres

Paramètre	Définition
base	Valeur numérique à multiplier par elle-même *exponent* nombre de fois.
exponent	Valeur numérique qui indique le nombre de fois que *base* doit être multipliée par elle-même. *exponent* doit être positif si *base* ≥ 0, et *exponent* doit être un entier si *base* < 0.

Description

La fonction pow() renvoie la valeur de la base portée à la puissance de l'entier, ou l'entier de base.

La plage de résultat pour la fonction pow() est $-\infty < pow() < \infty$.

printf()

Imprimer le texte structuré dans la bibliothèque C de la fonction printf().

Format

printf(*format*)

Paramètre

Paramètre	Définition
format	Texte, noms de variables et caractères de contrôle qui spécifient le contenu et le format des résultats à l'ordinateur ou à la fenêtre de résultat de tâche. format permet les caractères de conversion suivants: %d décimale signée (identique à %i) %i décimale signée (identique à %d) %u décimale non signée %o octal non signé %x hexadécimale non signée utilisant abcedf %X décimale non signée utilisant ABCDEF %c caractère non signé %s chaîne de caractères %e forme en double précision drddde±ddd où chaque d est un chiffre et r est le caractère de base %E forme de précision double drdddE±ddd où chaque d est un chiffre et r est le caractère de base %f forme de notation décimale dddrddd où chaque d est un chiffre et r est le caractère de base %g afficher dans le style de %e si l'entier après conversion est inférieur à –4, sinon, imprimer dans style %f %G afficher dans le style de %E avec la précision spécifiant le nombre de chiffres significatifs %N grouper les chiffres par 3 et les séparer par des virgules en commençant à droite de la chaîne %% afficher un caractère % format ne prend pas en charge les caractères de conversion du pointeur %p et %n du pointeur C standard. format permet les indicateurs suivants: justifier à gauche et remplir à droite avec des espaces + afficher signe plus lorsque la valeur est supérieure à zéro 0 remplir avec des zéros si aucun autre remplissage n'est spécifié # modifie la signification d'une conversion: ajoute 0x ou 0X aux conversions %x et %X ajoute toujours le caractère de base aux conversions %e, %E, %f, %g, et %G retient des zéros à droite dans les conversions %g et %G Le numéro de l'indicateur n'affecte pas les conversions %c, %d, %s ou %i

Description

La fonction printf() affiche les résultats à l'ordinateur ou à la fenêtre de résultats de tâches en utilisant une structure semblable à celle de la fonction printf() standard.

Un mauvais format ou un format qui est valide pour un langage C mais non pas pour la fonction printf() donne lieu à une erreur à l'exécution SMSL qui sélectionne la variable errno SMSL.

La valeur de retour de la fonction printf() est toujours la chaîne NULL.

Conventions C non prises en charge par la fonction SMSL printf

La fonction printf() ne prend pas en charge la convention C qui utilise l'astérisque (*) comme une largeur de champ ou un indicateur de précision. La fonction printf() ne prend pas en charge les caractères de conversion %p et %n.

Les modificateurs de longueur h, l (*ell*), et L ne sont pas valides et sont ignorés par la fonction printf.

Les conversions de format de la fonction printf() sont directement passées au programme printf() de la bibliothèque C sur chaque plate-forme. Les résultats des fonctions de mise en forme cachée peuvent être différents en fonction des plates-formes.

Différences de conversions entre le programme C printf et la fonction printf SMSL

Les conversions de format ont la même signification entre le standard C et SMSL mais le concept des types de variable est différent.

Le SMSL prend uniquement en charge les types de chaînes pour ses variables et, donc, les arguments de chaîne à la fonction printf() sont convertis de manière correcte pour la conversion de format:

- Les formats d'entier tels que %d convertissent la chaîne en entiers signés.
- Les formats numériques non d'entiers tels que %f convertissent en valeurs de virgule flottante.
- %c affiche l'équivalent ASCII de son argument d'entier ou des arguments non numériques pour le premier caractère de son argument. (%c à "65" affichera 'A' et à "AB" affichera 'A'.)
- %s n'entraîne aucune conversion.
- %% n'exige aucun argument.

La conversion de format %N

La fonction printf() fournit une extension C non standard – la conversion %N. La conversion %N prétraite une chaîne numérique afin de permettre aux virgules de séparer chaque groupe de trois chiffres en commençant à droite de la chaîne.

Par exemple, la conversion %N entraîne les conversions suivantes:

$$1234 \Rightarrow 1{,}234\ 12345 \Rightarrow 12{,}345\ 123456 \Rightarrow 123{,}456$$

Les conversions %N ne tiennent pas compte des moins et des espaces en cherchant la première séquence numérique pour que %N puisse être appliqué aux valeurs négatives. Si aucun chiffre n'est trouvé après les caractères sautés, l'argument affiché ne change pas.

La conversion %N modifie seulement la première séquence numérique. Par exemple, la conversion %N modifie des nombres à virgule flottante tels que 1234.1234 à 1,234.1234 sans modifier la séquence numérique à droite du point décimal.

Faisant partie de la conversion %N, la fonction printf() exécute une conversion %s en utilisant la largeur de champ et des spécificateurs de précision fournis dans le format. La fonction printf() affiche la chaîne qui résulte des deux conversions %N et %s. Du fait de la conversion %s intégrée, la largeur du champ et la précision sous la conversion %N ont le même effet que %s.

NOTE 7 – Aucune localisation n'est actuellement prise en charge par %N et, par conséquent, la mise en forme réalisée par %N ne change pas dans les différents lieux.

proc_exists()

Vérifier qu'un processus existe.

Format

proc_exists(*pid*)

Paramètre

Paramètre	Définition
pid	Numéro d'identificateur du processus dont l'existence est vérifiée

Description

La fonction proc_exists() renvoie VRAI si le processus avec le pid d'identificateur de processus existe; FAUX s'il n'existe pas.

process()

Renvoyer une liste de processus de l'antémémoire de processus de l'agent.

Format

process(*regular_expression*)

Paramètre

Paramètre	Définition
regular_expression	Séquence de caractères qui définit la configuration que la fonction process() recherche dans l'antémémoire de processus de l'agent. *regular_expression* est conforme aux expressions normales définies dans la description de commande UNIX ed(1) et la description UNIX regexp(5). Un résumé de plusieurs caractères d'expression normale est donné ci-dessous: ^ début de ligne \< début d'un mot $ fin de ligne \> fin d'un mot. Correspond à un seul caractère * correspond à zéro ou à plusieurs répétitions du précédent [] correspond à n'importe lequel des caractères contenus à l'intérieur [^] correspond à n'importe quels caractères à l'exception de ceux contenus à l'intérieur

Description

La fonction process() renvoie la liste de processus de l'antémémoire de processus de l'agent qui correspond à l'expression normale regular_expression. Chaque entrée dans la liste est une chaîne structurée comme suit:

pid ppid user status size cputime *command_name command_line*

NOTE 8 – Certaines plates-formes ne prennent pas en charge toutes les valeurs de retour. Pour une plate-forme spécifique, la fonction process() renvoie généralement la même information que la commande ps. La fonction process() renvoie la chaîne NULL si aucun processus ne correspond à regular_expression.

Exemple

Les commandes SMSL suivantes établissent une liste de tous les démons de processus de la base de données ORACLE.

```
# find ORACLE database daemons
ora_procs = process("ora_");
printf ("%d", ora_procs);
```

Paramètres

Paramètre	Définition
pid	Numéro de l'identificateur du processus
ppid	Numéro de l'identificateur du processus composé
user	Nom de l'utilisateur auquel le processus appartient
status	Etat du processus dans le système. Plage valide: 0 non existant S en sommeil W en attente R en fonctionnement I intermédiaire Z terminé T arrêté X en évolution
size	Taille de l'image mémoire du processus (dans les blocs)
cputime	Nombre entier des secondes UC consommées par le processus
command_name	Premier mot de la ligne de commande qui a lancé le processus
command_line	Ligne de commande complète qui a lancé le processus. Noter que la ligne de commande peut avoir été modifiée pendant l'exécution du processus.

random()

Renvoyer un nombre aléatoire.

Format

random(*maximum*)

Paramètre

Paramètre	Définition
maximum	Plage valide: maximum > 0 Par défaut si non spécifié: maximum = $2^{32} - 1$ (de la fonction C sous-jacente)

Description

La fonction random() est équivalente à la fonction random() de la bibliothèque standard C.

Si maximum est zéro ou négatif, la fonction random() renverra un message d'erreur à l'exécution. Limite supérieure optionnelle pour les valeurs renvoyées par le nombre aléatoire:

$$0 \leq \text{random}() \leq \text{maximum} - 1$$

read()

Lire à partir d'un fichier SMSL ou canal de traitement.

Format

read(*channel*,*size*)

Paramètres

Paramètre	Définition
channel	Numéro du canal E/S du processus pour lequel la fonction read() doit lire des données
size	Valeur d'entier contrôlant la quantité de données que la fonction read() lira sur le canal. Plage valide: size > 0 indique à la fonction read() de lire au moins six octets et retour size = 0 indique à la fonction read() de renvoyer dès qu'elle a lu quelque chose sur le canal size = –1 indique à la fonction read() de lire toutes les données disponibles sur le canal et retour Par défaut si non spécifié: size = 0

Description

La fonction read() renvoie les données qu'elle lit sur le canal. La fonction read() renvoie la valeur EOF (qui est la chaîne NULL) sur une fin de fichier ou une condition d'erreur.

Les canaux sont créés en appelant la fonction fopen() ou popen().

NOTE 9 – La fonction read peut bloquer pour un canal de processus créé en utilisant la fonction popen() mais non pour un canal de fichier créé en utilisant la fonction fopen().

Pour réaliser la sérialisation pour des canaux partagés, deux processus du lecteur (qui sont les fonctions read() ou readln()) ne peuvent être bloqués sur le même canal. Le second processus du lecteur qui tente de bloquer sur les canaux partagés échouera, en renvoyant la chaîne NULL et en positionnant la variable errno du SMSL à E_SMSL_BUSY_CHANNEL.

Un autre échec de canal partagé possible peut être dû à une fonction close() exécutée avec un canal qui a également un processus du lecteur bloqué. La fonction close() forcera le processus du lecteur à renvoyer la chaîne NULL et positionnera errno à E_SMSL_UNBLOCKED_BY_CLOSE.

Exemple

L'exemple SMSL suivant ouvre un canal au système d'exploitation UNIX, exécute une commande ls d'UNIX, puis lit et affiche les entrées de l'annuaire renvoyées par la commande ls.

```
chan = popen ("OS", "ls");
while ((data = read (chan)) ! = EOF)
{
    printf("%s", data);
}
close (chan);
```

readln()

Lire une ligne de données d'un fichier SMSL ou du canal de traitement.

Format

readln(*channel*)

Paramètre

Paramètre	Définition
channel	Numéro du canal E/S du processus pour lequel la fonction readln doit lire des données

Description

La fonction readln() lit la ligne suivante de données du canal et la renvoie. La fonction readln() renvoie la valeur EOF (NULL) sur une fin de fichier ou une erreur.

Les canaux sont créés en appelant la fonction fopen() ou popen(). Note: La fonction readln() peut bloquer pour un canal de communication en utilisant la fonction popen() mais non pour un canal de fichier créé en utilisant la fonction fopen().

Pour réaliser la sérialisation pour des canaux partagés, deux processus du lecteur (qui sont les fonctions read() ou readln()) ne peuvent être bloqués sur le même canal. Le second processus du lecteur qui tente de bloquer sur les canaux partagés échouera, en renvoyant la chaîne NULL et en positionnant la variable errno du SMSL à E_SMSL_BUSY_CHANNEL.

Un autre échec de canal partagé possible peut être dû à une fonction close() exécutée avec un canal qui a également un processus du lecteur bloqué. La fonction close() forcera le processus du lecteur à renvoyer la chaîne NULL et positionnera errno à E_SMSL_UNBLOCKED_BY_CLOSE.

Limitation

La fonction readln() a une limitation de ligne de 4K lorsqu'elle est exécutée avec des fichiers ouverts avec la fonction fopen(). La fonction readln() peut tronquer des lignes plus longues que 4K. Cette limitation ne s'applique pas aux canaux ouverts à l'aide de la fonction popen().

rindex()

Renvoyer la dernière apparition d'une chaîne de texte dans une autre.

Format

rindex(*text,string*)

Paramètres

Paramètre	Définition
text	Texte à examiner pour des apparitions de chaîne. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
string	Un ou plusieurs caractères dont la dernière apparition est en cours d'identification dans le texte.

Description

La fonction rindex() renvoie la position de la dernière apparition de chaîne dans le texte ou 0 si la chaîne n'apparaît pas dans le texte. Les positions dans la chaîne sont numérotées en commençant à partir de un.

set()

Attribuer une valeur à une variable.

Format

set(*variable,value*)

Paramètres

Paramètre	Définition
variable	Nom d'une variable dans la hiérarchie d'objet d'agent auquel la valeur est attribuée
value	Valeur numérique ou valeur de la chaîne qui est attribuée à la variable

Description

La fonction set() sélectionne la valeur de la variable à évaluer. Si la variable est un nom relatif et n'existe pas dans le contexte du script SMSL, la fonction set() recherche successivement chaque contexte hiérarchique jusqu'à ce qu'elle trouve la variable ou jusqu'à ce que la recherche échoue dans le contexte de l'ordinateur.

La fonction set() renvoie la valeur si l'attribution est réussie, la chaîne NULL en cas d'échec.

Exemple

L'instruction SMSL suivante positionne la valeur de la base de données RDB Dev parameter MyParam à 10.

```
set("/RDB/Dev/MyParam/value",10);
```

share()

Convertir un canal local en un canal global partagé.

Format

share(*channel,name*)

Paramètres

Paramètre	Définition
channel	Numéro du canal E/S de processus renvoyé lorsque le canal a été ouvert à l'aide de la fonction fopen() ou popen()
name	Chaîne de caractères utilisée pour identifier le canal partagé dans la table des canaux globaux. Il est recommandé de spécifier un nom non numérique pour éviter des conflits avec les nombres utilisés en interne pour les canaux locaux. Le fait d'utiliser un nombre pour le nom n'entraîne actuellement pas la fonction share() mais déclenche un avertissement à l'exécution du SMSL. La fonction share() placera respectivement le nom numérique spécifié dans la table globale, entraînant des conflits potentiels avec les canaux locaux dans les fonctions close(), read(), write() et readln().

Description

La fonction share() est la fonction principale à utiliser pour les canaux partagés. La fonction share() diffuse un canal local existant dans la table des canaux globaux comme un nom. Les canaux ouverts par la fonction popen() ou la fonction fopen() peuvent être partagés.

Si la fonction share() est réussie, elle renvoie à 0. Le canal local n'est plus disponible dans le processus qui l'a ouvert et ne nécessite pas une fonction close(). En fait, la fonction close() renverra une erreur puisqu'elle ne trouvera pas le canal local.

La fonction share() échouera, en renvoyant –1 et en sélectionnant la variable errno de SMSL si:

- le canal local n'existe pas;
- le nom du canal global existe déjà dans la table du canal global.

En cas d'échec, le canal local est inchangé et reste disponible. Aucun canal global n'est ajouté.

Un canal global est désigné par un nom lorsqu'il est transféré aux fonctions read(), readln(), write() et close(). Ces fonctions rechercheront, tout d'abord, la table de canal local contenant uniquement les numéros de canal et ensuite la table de canal global.

sin()

Renvoyer le sinus de l'argument.

Format

sin(*radians*)

Paramètre

Paramètre	Définition
radians	Longueur d'arc en radians dont le sinus doit être déterminé Plage valide: $-\infty \leq$ radians $\leq \infty$

Description

La fonction sin() renvoie le sinus des radians. La plage de résultat pour la fonction sin() est $-1 \leq \sin() \leq 1$.

sinh()

Renvoyer le sinus hyperbolique de l'argument.

Format

sinh(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur numérique dont le sinus hyperbolique doit être déterminé Plage valide: $-\infty \leq$ argument $\leq \infty$

Description

La fonction sinh() renvoie le sinus hyperbolique de l'argument. Le sinus hyperbolique est défini par l'expression:

$$\sinh(x) = (e^{x} - e^{-x})/2$$

où *e* est la base pour les logarithmes naturels (e = 2,71828 . . .). La plage de résultat pour la fonction sinh() est $-\infty \leq \sinh() \leq \infty$.

sleep()

Suspendre l'exécution du processus pendant un nombre de secondes spécifié.

Format

sleep(*seconds*)

Paramètre

Paramètre	Définition
seconds	Entier spécifiant le nombre de secondes de mise en suspension du processus. Plage valide: seconds > 0 est le nombre de secondes de mise en sommeil du processus seconds ≤ 0 la temporisation prend fin immédiatement

Description

La fonction sleep() suspend un processus SMSL pendant le nombre de secondes spécifié. Tandis qu'il est suspendu, le processus SMSL ne consomme aucune ressource UC et n'est pas interprété avant d'être "réveillé" par l'expiration de la temporisation en secondes.

NOTE 10 – La fonction sleep() suspend uniquement le processus qu'elle a appelé. Tous les autres processus SMSL continuent l'exécution normalement.

La fonction sleep() renvoie un avertissement à l'exécution si les secondes sont non numériques, dans ce cas, la temporisation par défaut est à zéro.

sort()

Trier une liste de valeurs numériques ou alphabétiques.

Format

sort(*list*,*mode*,*position*)

Paramètres

Paramètre	Définition
list	Liste SMSL dont les éléments doivent être triés
mode	Chaîne de caractères optionnelle spécifiant l'ordre de tri. La chaîne de caractères doit être entre guillemets. Plage valide: "n" ordre numérique croissant; "nr" ordre numérique décroissant; "" ordre alphabétique croissant; "r" ordre alphabétique décroissant. Par défaut si non spécifié: "" (alphabétique croissant).
position	Entier optionnel qui spécifie la position du caractère dans chaque élément de la liste où le tri doit commencer. Le premier caractère de chaque élément de liste est le caractère 1. Si la longueur de chaque élément dans la liste est inférieure à la position, l'effet est le même que si tous les éléments de la liste sont des éléments NULL. position ne tronque pas les éléments; elle ignore uniquement les premiers (position –1) caractères à des fins de comparaison. Par défaut si non spécifié: 1.

Description

La fonction sort() renvoie une version triée de la liste qui est classée conformément au mode.

La fonction sort() ne fusionne pas les entrées dupliquées dans la liste: la liste renvoyée a le même nombre d'éléments que la liste. L'ordre dans lequel les doubles sont renvoyés n'est pas défini car il n'est pas défini pour la fonction qsort() de la bibliothèque C. Ceci est particulièrement significatif dans les cas suivants:

- tri numérique de chaînes avec des valeurs de préfixe numériques identiques mais des suffixes non numériques différents;
- tout mode de tri dans lequel la position est plus grande de plus d'un élément dans la liste (la fonction sort() traite tous ces éléments comme des éléments dupliqués NULL).

Si la liste est la liste NULL, la fonction tri renvoie la liste NULL. Pour une liste non vide, la fonction sort() renvoie toujours une liste bien définie avec la dernière ligne se terminant par un caractère d'interligne.

NOTE 11 – Un caractère d'interligne n'est pas nécessaire pour terminer la liste. Le tri numérique est fondé sur les valeurs de virgule flottante; les entrées de liste non numériques sont converties conformément à la fonction atof() de la bibliothèque C standard.

sprintf()

Renvoyer le format spécifié comme une chaîne de caractères à une destination.

Format

sprintf(*format*)

Paramètre

Paramètre	Définition
format	Texte, noms de variable et caractères de commande qui spécifient le contenu et le format de la sortie de la chaîne de caractères à l'ordinateur ou à la fenêtre de résultat de la tâche. Le format permet les caractères de conversion suivants: %d décimale signée (identique à %i) %i décimale signée (identique à %d) %u décimale non signée %o octal non signé %x hexadécimale non signée utilisant abcedf %X décimale non signée utilisant ABCDEF %c caractère non signé %s chaîne de caractères %e forme de précision double drddde±ddd où chaque d est un chiffre et r est le caractère de base %E forme de précision double drdddE±ddd où chaque d est un chiffre et r est le caractère de base %f forme de notation décimale dddrddd où chaque d est un chiffre et r est le caractère de base %g affiche dans le style de %e si l'exposant après conversion est inférieur à –4, sinon affiche dans le style %f %G affiche dans le style de %E avec la précision spécifiant le nombre de chiffres significatifs %N groupe les chiffres par trois et les sépare par des virgules en commençant à droite de la chaîne %% affiche un caractère % format ne prend pas en charge les caractères de conversion %p et %n du pointeur C standard. format permet les indicateurs suivants: – justifier à gauche et remplir à droite avec des espaces + afficher le signe plus lorsque la valeur est supérieure à zéro 0 remplir avec des zéros si aucun autre remplissage n'est spécifié # modifie la signification d'une conversion: ajoute 0x ou 0X aux conversions %x et %X ajoute toujours le caractère de base aux conversions %e, %E, %f, %g et %G retient les zéros à droite dans les conversions %g et %G Le numéro de l'indicateur # n'affecte pas les conversions %c, %d, %s ou %i.

Description

La fonction sprintf() est identique à la fonction printf() à l'exception du fait que la chaîne créée est renvoyée et non mise en sortie. S'il n'y a pas d'erreur dans le format, sprintf sélectionne la variable errno de SMSL et renvoie la chaîne NULL.

Les formats, conversions et valeurs de errno pour les différentes erreurs sont identiques à ceux décrits pour la fonction printf().

Les programmeurs C doivent utiliser avec précaution le style SMSL:

destination=sprintf(*format*)

plutôt que le style C:

sprintf(*destination*,*format*)

Ce style donnera plus souvent lieu à un avertissement de compilation concernant l'instruction à effet nul.

Conventions C non prises en charge par la fonction sprintf SMSL

La fonction sprintf() ne prend pas en charge la convention C qui utilise l'astérisque (*) comme une largeur de champ ou un indicateur de précision. La fonction sprintf() ne prend pas en charge les caractères de conversion %p et %n.

Les modificateurs de longueur h, l (*ell*), et L ne sont pas valides et sont ignorés par la fonction sprintf().

Les conversions de format de la fonction sprintf() sont directement passées au programme sprintf() de la bibliothèque C sur chaque plate-forme. Le résultat pour les fonctions de mise en forme cachée peut être différent selon les plates-formes.

Différences de conversion entre le programme C sprintf et la fonction sprintf du SMSL

Les conversions de format ont la même signification entre le programme C standard et le SMSL mais le concept des types de variables est différent.

Le SMSL prend en charge uniquement les types de chaîne pour ses variables et, par conséquent, les arguments de chaîne à la fonction sprintf() sont convertis de façon appropriée pour la conversion de format:

- Les formats d'entier tels que %d convertissent la chaîne en entiers signés.
- Les formats numériques non d'entier tels que %f convertissent les valeurs de la virgule flottante.
- %c affiche l'équivalent ASCII de ses arguments d'entier ou d'arguments non numériques, le premier caractère de son argument. (%c à "65" affichera 'A' et à "AB" affichera 'A'.)
- %s n'entraîne pas de conversion.
- %% ne nécessite pas d'argument.

sqrt()

Renvoyer la racine carrée de l'argument.

Format

sqrt(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur numérique dont la racine carrée mathématique est renvoyée Plage valide: $-\infty \leq$ argument $\leq \infty$

Description

La fonction sqrt() renvoie la racine carrée de l'argument de valeur positive d'entier ou réelle.

srandom()

Initialise le générateur de nombre aléatoire avec une valeur de départ.

Format

srandom(*seed*)

Paramètre

Paramètre	Définition
seed	Valeur numérique utilisée comme un point initial pour la génération de nombre pseudo-aléatoire par la fonction random()

Description

La fonction srandom() sélectionne la valeur de départ du nombre aléatoire pour la fonction random(). La valeur de départ est directement passée à la fonction srandom() d'UNIX C.

La fonction srandom() SMSL renvoie toujours la chaîne NULL.

subset()

Vérifier qu'une liste SMSL est un sous-ensemble d'une autre.

Format

subset(*set*,*subset*)

Paramètres

Paramètre	Définition
set	Liste SMSL qui est l'ensemble dans la vérification ensemble-sous-ensemble
subset	Liste SMSL qui est le sous-ensemble dans la vérification ensemble-sous-ensemble

Description

La fonction subset() renvoie une valeur booléenne de 0 ou de 1 en indiquant si le sous-ensemble est un sous-ensemble correct ou incorrect de l'ensemble. Si le sous-ensemble est l'ensemble NULL, la fonction subset() renvoie 1 (VRAI). Si l'ensemble est l'ensemble NULL et le sous-ensemble ne l'est pas, la fonction subset() renvoie 0 (FAUX).

La fonction subset() ignore les doubles et renvoie 1 uniquement si tous les éléments du sous-ensemble sont aussi présents dans l'ensemble.

Exemple

La fonction subset() peut être utilisée pour déterminer si un élément particulier est présent dans un ensemble et fournit alors la fonction "is_member" comme suit:

```
if (subset(my_set,"blue"))
{
        # SMSL set "my_set" contains element "blue"
}
```

Il n'est pas nécessaire de mettre un interligne à la fin de la chaîne "blue" car elle est insérée par la fonction subset(). Les instructions d'exemple sont traitées comme une fonction subset() agissant sur un ensemble à un élément.

substr()

Renvoyer une partie spécifiée d'une chaîne de caractères.

Format

substr(*text*,*start*,*length*)

Paramètres

Paramètre	Définition
text	Texte à partir duquel une sous-chaîne de caractères n'est pas renvoyée. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
start	Position du caractère dans le texte qui doit être le premier caractère de la sous-chaîne. Le premier caractère dans le texte est la position de caractère 1.
length	Nombre total de caractères du texte à renvoyer dans la sous-chaîne

Description

La fonction substr() renvoie la sous-chaîne de texte de longueur de caractères qui démarre à la position *start*.

system()

Transmettre une commande au système d'exploitation de l'ordinateur.

Format

system(*command*,*instance*)

Paramètres

Paramètre	Définition
command	Syntaxe de la commande du système d'exploitation transmise. *command* peut contenir la réorientation de sortie, les canaux de communication, les jokers, etc.
instance	Instance d'application optionnelle avec laquelle la commande doit exécuter. Par défaut si non spécifié: L'instance d'application hiérarchiquement la plus proche de la commande

Description

La fonction system() renvoie tout résultat produit par la commande d'émission au sous-système d'exécution de la commande dépendant du système.

tail()

Renvoyer les dernières lignes d'un bloc de texte.

Format

tail(*text*,*lines*)

Paramètres

Paramètre	Définition
text	Texte dont les dernières lignes doivent être renvoyées par tail(). *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
lines	Nombre de lignes du texte à renvoyer en commençant par la première ligne de texte

Description

La fonction tail() renvoie le dernier nombre de lignes des lignes de texte.

tan()

Renvoyer la tangente de l’argument.

Format

tan(*radians*)

Paramètre

Paramètre	Définition
radians	Longueur de l'arc en radians dont la tangente est à déterminer Plage valide: $-\infty \leq$ radians $\leq \infty$

Description

La fonction tan() renvoie la tangente des radians. La plage de résultat pour la fonction tan() est $-\infty < \tan() < \infty$. La fonction tan() est indéfinie lorsque les radians = p(2n+1)/2 où n est un entier.

tanh()

Renvoyer la tangente hyperbolique de l'argument.

Format

tanh(*argument*)

Paramètre

Paramètre	Définition
argument	Valeur numérique dont la tangente hyperbolique est à déterminer Plage valide: $-\infty \leq$ argument $\leq \infty$

Description

La fonction tanh() renvoie la tangente hyperbolique de l'argument. La tangente hyperbolique est définie par l'expression:

$$\tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$$

où e est la base pour les logarithmes naturels (e = 2,71828 . . .). La plage de résultat pour la fonction tanh() est $-1 \leq \tanh() \leq 1$.

time()

Renvoyer le nombre de secondes depuis 00:00:00 GMT 1er janvier 1970.

Format

time()

Description

La fonction time() renvoie l'heure actuelle comme le nombre de secondes qui se sont écoulées depuis 00:00:00 GMT, 1er janvier 1970.

tmpnam()

Renvoyer un nom unique pour la création de fichiers temporaires.

Format

tmpnam()

Description

La fonction tmpnam() renvoie un nom qui est garanti unique et peut être utilisé pour passer à la fonction fopen pour la création de fichiers temporaires.

La sémantique de la fonction tmpnam() est similaire à celles du programme C tmpnam() – notamment, un nombre limité de noms uniques est renvoyé par le programme tmpnam() comme défini par la constante C TMP_MAX. Tous les processus SMSL sur un agent donné partagent le même ensemble de noms, ce qui peut représenter un risque pour les noms associés. Si la taille de TMP_MAX pose problème, ajouter un suffixe au nom du fichier renvoyé.

Exemple

Les exemples suivants montrent comment utiliser la fonction tmpnam() pour générer un nom de fichier temporaire. La fonction SMSL ajoute un suffixe au nom renvoyé pour en garantir ultérieurement son unicité.

```
name = tmpnam() . ".dave"; fp = fopen(name, "w");
```

tolower()

Convertir le texte en caractères minuscules.

Format

tolower(*text*)

Paramètre

Paramètre	Définition
text	Texte à renvoyer en lettres minuscules. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.

Description

La fonction tolower() renvoie une copie du texte avec toutes les lettres majuscules converties en lettres minuscules.

toupper()

Convertir le texte en caractères majuscules.

Format

toupper(*text*)

Paramètre

Paramètre	Définition
text	Texte à renvoyer en lettres majuscules. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.

Description

La fonction toupper() renvoie une copie du texte avec toutes les lettres minuscules converties en lettres majuscules.

trim()

Retirer les caractères non désirés du texte.

Format

trim(*text,unwanted*)

Paramètres

Paramètre	Définition
text	Texte à renvoyer sans caractères spécifiés. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.
unwanted	Un ou plusieurs caractères à retirer de la copie du texte sorti par la fonction trim.

Description

La fonction trim() renvoie une copie du texte avec toutes les apparitions des caractères retirés non désirés.

union()

Renvoyer une liste qui est la réunion des listes individuelles.

Format

union(*list1,list2,list3,list4 . . . listn*)

Paramètres

Paramètre	Définition
listn	Liste SMSL contenant les éléments à associer et à renvoyer dans une seule liste bien définie. Seules les deux premières listes d'entrée, list1 et list2, sont exigées; toutes les autres sont en option.

Description

La fonction union() renvoie une liste SMSL qui contient les éléments de toutes les listn fusionnées. A la différence des fonctions difference() et intersection(), la liste renvoyée par la fonction union est un ensemble bien défini sans aucun double. La fonction union() ajoute un nouvel interligne à la fin de chaque liste non vide qui manque. Si la valeur de retour n'est pas la liste NULL, l'ensemble renvoyé se termine toujours par un interligne afin que tous les éléments de l'ensemble se terminent par un caractère d'interligne.

unique()

Retirer les éléments en double d'une liste.

Format

unique(*list*)

Paramètre

Paramètre	Définition
list	Liste SMSL contenant les éléments à renvoyer dans une seule liste (*unique*) bien définie

Description

La fonction unique() renvoie une liste SMSL bien définie avec tous les doubles enlevés. Tous les éléments qui restent dans la valeur de retour apparaissent dans le même ordre dans lequel ils figuraient dans la liste. Si la liste est la liste NULL, la fonction unique() renvoie la liste NULL; autrement, la fonction unique() renvoie une liste qui se termine par un caractère d'interligne afin que tous les éléments de liste dans la liste se terminent par un caractère d'interligne.

unlock()

Libérer un verrou de processus SMSL.

Format

unlock(*lockname*)

Paramètre

Paramètre	Définition
lockname	Nom du verrou à libérer

Description

La fonction unlock() libère *lockname* qui avait été accordé à ce processus par un appel précédent à la fonction lock(). La fonction unlock() renvoie 1 en cas de réussite et 0 en cas d'échec. Si aucun verrou n'est nommé *lockname* ou s'il n'est pas actuellement détenu par ce processus, la fonction unlock() signale alors une erreur à l'exécution, sélectionne la variable errno de SMSL et renvoie 0.

NOTE 12 – Tous les verrous détenus par un processus sont automatiquement libérés quand le processus quitte d'une manière similaire pour exécuter la fonction unlock. Il est recommandé de libérer les verrous de manière explicite à l'aide de la fonction unlock() plutôt que de manière implicite à l'aide du processus exit. Si ce processus est le seul à détenir le verrou et que les processus sont en file d'attente, le premier processus en attente est "réveillé" et l'utilisation du verrou est accordée. Si le premier processus est une demande partagée, tout autre processus qui est en file d'attente pour un verrou partagé est également autorisé à partager le verrou (sauf pour les processus qui sont derrière une demande d'écriture sur la file d'attente pour ce verrou).

write()

Ecrire un processus SMSL ou un canal de fichier.

Format

write(*chan*,*text*)

Paramètres

Paramètre	Définition
chan	Numéro de canal E/S de processus pour lequel *text* est écrit
text	Texte à écrire au canal *chan*. *text* peut être une chaîne de texte entre guillemets ou une ou plusieurs commandes SMSL qui produisent un texte en résultat.

Description

La fonction write() écrit le texte au canal chan. La fonction write() renvoie le nombre de caractères écrits ou –1 en erreur.

Si le texte ne peut pas être écrit immédiatement, la fonction write() appelle les blocs jusqu'à ce qu'elle puisse écrire tout le texte ou que le canal se termine.

NOTE 13 – La fonction write() peut bloquer pour un canal de processus créé en utilisant la fonction popen() mais non pour un canal de fichier créé en utilisant la fonction fopen().

Pour réaliser la sérialisation pour des canaux partagés, deux processus du lecteur (qui sont les fonctions read() ou readln()) ne peuvent pas être bloqués sur le même canal. Le second processus du lecteur qui tente de bloquer sur le canal partagé échouera, en renvoyant la chaîne NULL et en positionnant la variable errno du SMSL à E_SMSL_BUSY_CHANNEL.

Un autre échec de canal partagé possible peut être dû à une fonction close() exécutée avec un canal qui a également un processus du lecteur bloqué. La fonction close() forcera le processus du lecteur à renvoyer la chaîne NULL et positionnera errno à E_SMSL_UNBLOCKED_BY_CLOSE.

Annexe H

Formulaire de déclaration de conformité d'objet géré MOCS

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

This annex contains MOCS proforma for the subset of object classes defined in [X721] that is used for SDH management.

The following common notations, defined in Recommendation X.724 are used for the status columns:

- m Mandatory
- o Optional
- c Conditional
- x Prohibited
- – Not applicable or out of scope

Note that “c”, “m”, “o” and “x” are prefixed by a “c:” when nested under a conditional or optional item of the same table.

Note that “o” may be suffixed by “n” (where “n” is a unique number) for mutually exclusive or selectable options among a set of status values.

In the status column, the static requirements are stated as follows:

- m For characteristics contained in mandatory packages or in conditional packages if the GDMO condition is always true.
- o For characteristics of conditional packages with GDMO conditions that indicate static optionality, e.g. “if an instance supports it”.
- cn For all other conditions, where “n” is a unique integer and “cn” is a reference to a conditional status expression as defined in ITU-T Rec. X.291 | ISO/IEC 9646-2 and ITU-T Rec. X.296 | ISO/IEC 9646-7. Each condition denoted by “cn” is relative to the containing table.
- x For characteristics explicitly prohibited by the definition.
- – For characteristics that are not mentioned by the definition.

The following common notations, defined in ITU-T Rec. X.724 | ISO/IEC 10165-6 and Rec. X.296 | ISO/IEC 9646-7 are used for the support answer columns:

- Y Implemented
- N Not implemented
- – No answer required

The following abbreviations are used:

smi2AttributeID	{ joint-iso-itu-t ms(9) smi(3) part2(2) attribute(7) }
smi2MObjectClass	{ joint-iso-itu-t ms(9) smi(3) part2(2) managedObjectClass(3) }
smi2Notification	{ joint-iso-itu-t ms(9) smi(3) part2(2) notification(10) }

H.1 Statement of conformance to the basicSpawnerClass object class

Table H.1 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	basicSpawnerClass	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx1(1)}		

If the answer to the actual class question in the managed object class support Table H.1 is no, the supplier of the implementation shall fill in the actual class support in Table H.2.

Table H.2 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.1.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.3.

Table H.3 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-1/1b) then m else –						

H.1.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.4. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.4 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	allomorphs	{smi2AttributeID 50}		x		c1		x	
2	nameBinding	{smi2AttributeID 63}		–		m		x	
3	objectClass	{smi2AttributeID 65}		–		m		x	
4	packages	{smi2AttributeID 66}		–		m		x	

Table H.4 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
c1: if not (H-1/1b) then m else –							

H.1.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.1.4 Actions

There are no actions defined for this object class.

H.1.5 Notifications

There are no notifications defined for this object class.

H.1.6 Parameters

There are no parameters defined for this object class.

H.2 Statement of conformance to the commandSequencer object class

Table H.5 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	commandSequencer	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx2(2)}		

If the answer to the actual class question in the managed object class support Table H.5 is no, the supplier of the implementation shall fill in the actual class support in Table H.6.

Table H.6 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.2.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.7.

Table H.7 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-5/1b) then m else – c2: if H-7/1 then m else –						

H.2.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.8. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.8 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	commandSequencerId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx2(2)}		–		m		x	
4	nameBinding	{smi2AttributeID 63}		–		m		x	
5	objectClass	{smi2AttributeID 65}		–		m		x	
6	operationalState	{smi2AttributeID 35}		–		m		x	
7	packages	{smi2AttributeID 66}		–		c2		x	

Table H.8 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		

c1: if not (H-5/1b) then m else –
c2: if H-7/2 then m else –

H.2.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.2.4 Actions

There are no actions defined for this object class.

H.2.5 Notifications

The supplier of the implementation shall state whether or not the notifications specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.9. The supplier of the implementation shall indicate support in terms of the confirmed and non-confirmed modes.

Table H.9 – MOCS – Notification support

Index	Notification type template label	Value of object identifier for notification type	Constraints and values	Status	Support		Additional information
					Confirmed	Non-confirmed	
1	objectCreation	{smi2Notification 6}		m			
2	objectDeletion	{smi2Notification 7}		m			
3	stateChange	{smi2Notification 14}		m			

Table H.9 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
1	1.1	additionalInformation	{smi2AttributeID 6}		o		
	1.1.1	identifier	–		c:m		
	1.1.2	significance	–		c:m		
	1.1.3	information	–		c:m		
	1.2	additionalText	{smi2AttributeID 7}		o		
	1.3	attributeList	{smi2AttributeID 9}		o		
	1.3.1	attributeId	–		c:m		
	1.3.1.1	globalForm	–		c:o.1		
	1.3.1.2	localForm	–		c:o.1		
	1.3.2	attributeValue	–		c:m		
	1.4	correlatedNotifications	{smi2AttributeID 12}		o		
	1.4.1	correlatedNotifications	–		c:m		
	1.4.2	sourceObjectInst	–		c:o		
	1.4.2.1	distinguishedName	–		c:o.2		
	1.4.2.1.1	AttributeType	–		c:m		
	1.4.2.1.2	AttributeValue	–		c:m		
	1.4.2.2	nonSpecificForm	–		c:o.2		
	1.4.2.3	localDistinguishedName	–		c:o.2		
	1.4.2.3.1	AttributeType	–		c:m		
	1.4.2.3.2	AttributeValue	–		c:m		
	1.5	notificationIdentifier	{smi2AttributeID 16}		o		
	1.6	sourceIndicator	{smi2AttributeID 26}		o		
2	2.1	additionalInformation	{smi2AttributeID 6}		o		
	2.1.1	identifier	–		c:m		
	2.1.2	significance	–		c:m		
	2.1.3	information	–		c:m		
	2.2	additionalText	{smi2AttributeID 7}		o		
	2.3	attributeList	{smi2AttributeID 9}		o		
	2.3.1	attributeId	–		c:m		
	2.3.1.1	globalForm	–		c:o.3		
	2.3.1.2	localForm	–		c:o.3		
	2.3.2	attributeValue	–		c:m		
	2.4	correlatedNotifications	{smi2AttributeID 12}		o		
	2.4.1	correlatedNotifications	–		c:m		
	2.4.2	sourceObjectInst	–		c:o		
	2.4.2.1	distinguishedName	–		c:o.4		
	2.4.2.1.1	AttributeType	–		c:m		
	2.4.2.1.2	AttributeValue	–		c:m		
	2.4.2.2	nonSpecificForm	–		c:o.4		
	2.4.2.3	localDistinguishedName	–		c:o.4		

Table H.9 *(concluded)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
	2.4.2.3.1	AttributeType	–		c:m		
	2.4.2.3.2	AttributeValue	–		c:m		
	2.5	notificationIdentifier	{smi2AttributeID 16}		o		
	2.6	sourceIndicator	{smi2AttributeID 26}		o		
3	3.1	additionalInformation	{smi2AttributeID 6}		o		
	3.1.1	identifier	–		c:m		
	3.1.2	significance	–		c:m		
	3.1.3	information	–		c:m		
	3.2	additionalText	{smi2AttributeID 7}		o		
	3.3	attributeIdentifierList	{smi2AttributeID 8}		o		
	3.3.1	globalForm	–		c:o.5		
	3.3.2	localForm	–		c:o.5		
	3.4	correlatedNotifications	{smi2AttributeID 12}		o		
	3.4.1	correlatedNotifications	–		c:m		
	3.4.2	sourceObjectInst	–		c:o		
	3.4.2.1	distinguishedName	–		c:o.6		
	3.4.2.1.1	AttributeType	–		c:m		
	3.4.2.1.2	AttributeValue	–		c:m		
	3.4.2.2	nonSpecificForm	–		c:o.6		
	3.4.2.3	localDistinguishedName	–		c:o.6		
	3.4.2.3.1	AttributeType	–		c:m		
	3.4.2.3.2	AttributeValue	–		c:m		
	3.5	notificationIdentifier	{smi2AttributeID 16}		o		
	3.6	sourceIndicator	{smi2AttributeID 26}		o		
	3.7	stateChangeDefinition	{smi2AttributeID 28}		m		
	3.7.1	attributeID	–		m		
	3.7.1.1	globalForm	–		c:o.7		
	3.7.1.2	localForm	–		c:o.7		
	3.7.2	oldAttributeValue	–		o		
	3.7.3	newAttributeValue	–		m		

H.2.6 Parameters

There are no parameters defined for this object class.

H.3 Statement of conformance to the generalStringScript object class

Table H.10 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	generalStringScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx3(3)}		

If the answer to the actual class question in the managed object class support Table H.10 is no, the supplier of the implementation shall fill in the actual class support in Table H.11.

Table H.11 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.3.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.12.

Table H.12 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-10/1b) then m else –						

H.3.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.13. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.13 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
4	nameBinding	{smi2AttributeID 63}		–		m		x	
5	objectClass	{smi2AttributeID 65}		–		m		x	
6	packages	{smi2AttributeID 66}		–		m		x	
7	scriptContent	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx4(4)}		m		m		m	
8	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
9	scriptLanguageName	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx7(7)}		m		m		m	

Table H.13 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
c1: if not (H-10/1b) then m else –							

H.3.4 Attribute groups

There are no attribute groups defined for the managed object class.

H.3.5 Actions

There are no actions defined for this object class.

H.3.6 Notifications

There are no notifications defined for this object class.

H.3.7 Parameters

There are no parameters defined for this object class.

H.4 Statement of conformance to the launchPad object class

Table H.14 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	launchPad	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx6(6)}		

If the answer to the actual class question in the managed object class support Table H.14 is no, the supplier of the implementation shall fill in the actual class support in Table H.15.

Table H.15 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.4.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.16.

Table H.16 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-14/1b) then m else –						

H.4.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.17. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.17 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		–		x		x	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	availabilityStatus	{smi2AttributeID 33}		–		x		x	
4	controlStatus	{smi2AttributeID 34}		–		m		x	
5	launchPadId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx6(6)}		–		m		x	
6	nameBinding	{smi2AttributeID 63}		–		m		x	
7	objectClass	{smi2AttributeID 65}		–		m		x	
8	observedAttributeId	{joint-iso-itu-t ms(9) function(2) part11(11) attribute(7) xx15(15)}		m		m		m	
9	observedObjectInstance	{joint-iso-itu-t ms(9) function(2) part11(11) attribute(7) xx16(16)}		m		m		m	
10	operationalState	{smi2AttributeID 35}		–		x		x	
11	packages	{smi2AttributeID 66}		–		m		x	
12	schedulerName	{smi2AttributeID 67}		–		m		x	
13	usageState	{smi2AttributeID 39}		–		x		x	

Table H.17 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
12	x		x		x		
13	x		x		x		
c1: if not (H-14/1b) then m else –							

H.4.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.4.4 Actions

The supplier of the implementation shall state whether or not the actions specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.18.

Table H.18 – MOCS – Action support

Index	Action type template label	Value of object identifier for action type	Constraints and values	Status	Support	Additional information
1	resume	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx1(1)}		m		
2	suspend	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx2(2)}		m		
3	terminate	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx3(3)}		m		

Table H.19 – MOCS – Action support

Index	Subindex	Action field name label	Constraints and values	Status	Support	Additional information
1	1.1	SpawnerObjectId		m		
	1.1.1	triggerId		m		
	1.1.1.1	distinguishedName		c:o.1		
	1.1.1.1.1	AttributeType		c:m		
	1.1.1.1.2	AttributeValue		c:m		
	1.1.1.2	nonSpecificForm		c:o.1		
	1.1.1.3	localDistinguishedName		c:o.1		
	1.1.1.3.1	AttributeType		c:m		
	1.1.1.3.2	AttributeValue		c:m		
	1.1.2	CHOICE		m		
	1.1.2.1	threadId		c:o.2		
	1.1.2.1.1	distinguishedName		c:o.3		
	1.1.2.1.1.1	AttributeType		c:m		
	1.1.2.1.1.2	AttributeValue		c:m		
	1.1.2.1.2	nonSpecificForm		c:o.3		
	1.1.2.1.3	localDistinguishedName		c:o.3		
	1.1.2.1.3.1	AttributeType		c:m		
	1.1.2.1.3.2	AttributeValue		c:m		
	1.1.2.2	launchPadId		c:o.2		
	1.1.2.2.1	distinguishedName		c:o.4		
	1.1.2.2.1.1	AttributeType		c:m		
	1.1.2.2.1.2	AttributeValue		c:m		
	1.1.2.2.2	nonSpecificForm		c:o.4		
	1.1.2.2.3	localDistinguishedName		c:o.4		
	1.1.2.2.3.1	AttributeType		c:m		
	1.1.2.2.3.2	AttributeValue		c:m		
2	2.1	SpawnerObjectId		m		
	2.1.1	triggerId		m		
	2.1.1.1	distinguishedName		c:o.5		
	2.1.1.1.1	AttributeType		c:m		
	2.1.1.1.2	AttributeValue		c:m		
	2.1.1.2	nonSpecificForm		c:o.5		
	2.1.1.3	localDistinguishedName		c:o.5		
	2.1.1.3.1	AttributeType		c:m		
	2.1.1.3.2	AttributeValue		c:m		
	2.1.2	CHOICE		m		
	2.1.2.1	threadId		c:o.6		
	2.1.2.1.1	distinguishedName		c:o.7		
	2.1.2.1.1.1	AttributeType		c:m		
	2.1.2.1.1.2	AttributeValue		c:m		
	2.1.2.1.2	nonSpecificForm		c:o.7		
	2.1.2.1.3	localDistinguishedName		c:o.7		
	2.1.2.1.3.1	AttributeType		c:m		
	2.1.2.1.3.2	AttributeValue		c:m		
	2.1.2.2	launchPadId		c:o.6		
	2.1.2.2.1	distinguishedName		c:o.8		
	2.1.2.2.1.1	AttributeType		c:m		
	2.1.2.2.1.2	AttributeValue		c:m		
	2.1.2.2.2	nonSpecificForm		c:o.8		
	2.1.2.2.3	localDistinguishedName		c:o.8		
	2.1.2.2.3.1	AttributeType		c:m		
	2.1.2.2.3.2	AttributeValue		c:m		
3	3.1	TriggerId		m		
	3.1.1	distinguishedName		c:o.9		
	3.1.1.1	AttributeType		c:m		
	3.1.1.2	AttributeValue		c:m		
	3.1.2	nonSpecificForm		c:o.9		
	3.1.3	localDistinguishedName		c:o.9		
	3.1.3.1	AttributeType		c:m		
	3.1.3.2	AttributeValue		c:m		

H.4.5 Notifications

There are no notifications defined for this object class.

H.4.6 Parameters

There are no parameters defined for this object class.

H.5 Statement of conformance to the asynchronousLaunchPad object class

Table H.20 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	asynchronousLaunchPad	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx4(4)}		

If the answer to the actual class question in the managed object class support Table H.20 is no, the supplier of the implementation shall fill in the actual class support in Table H.21.

Table H.21 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.5.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.22.

Table H.22 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-20/1b) then m else –						

H.5.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.23. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

H.5.3 Attribute groups

There are no attribute groups defined for the managed object class.

Table H.23 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		–		x		x	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	availabilityStatus	{smi2AttributeID 33}		–		x		x	
4	controlStatus	{smi2AttributeID 34}		–		m		x	
5	launchPadId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx6(6)}		–		m		x	
6	nameBinding	{smi2AttributeID 63}		–		m		x	
7	objectClass	{smi2AttributeID 65}		–		m		x	
8	observedAttributeId	{joint-iso-itu-t ms(9) function(2) part11(11) attribute(7) xx15(15)}		m		m		m	
9	observedObjectInstance	{joint-iso-itu-t ms(9) function(2) part11(11) attribute(7) xx16(16)}		m		m		m	
10	operationalState	{smi2AttributeID 35}		–		x		x	
11	packages	{smi2AttributeID 66}		–		m		x	
12	schedulerName	{smi2AttributeID 67}		–		m		x	
13	usageState	{smi2AttributeID 39}		–		x		x	

Table H.23 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
12	x		x		x		
13	x		x		x		
c1: if not (H-20/1b) then m else –							

H.5.4 Actions

The supplier of the implementation shall state whether or not the actions specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.24.

Table H.24 – MOCS – Action support

Index	Action type template label	Value of object identifier for action type	Constraints and values	Status	Support	Additional information
1	resume	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx1(1)}		m		
2	suspend	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx2(2)}		m		
3	terminate	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx3(3)}		m		

Table H.25 – MOCS – Action support

Index	Subindex	Action field name label	Constraints and values	Status	Support	Additional information
1	1.1	SpawnerObjectId		m		
	1.1.1	triggerId		m		
	1.1.1.1	distinguishedName		c:o.1		
	1.1.1.1.1	AttributeType		c:m		
	1.1.1.1.2	AttributeValue		c:m		
	1.1.1.2	nonSpecificForm		c:o.1		
	1.1.1.3	localDistinguishedName		c:o.1		
	1.1.1.3.1	AttributeType		c:m		
	1.1.1.3.2	AttributeValue		c:m		
	1.1.2	CHOICE		m		
	1.1.2.1	threadId		c:o.2		
	1.1.2.1.1	distinguishedName		c:o.3		
	1.1.2.1.1.1	AttributeType		c:m		
	1.1.2.1.1.2	AttributeValue		c:m		
	1.1.2.1.2	nonSpecificForm		c:o.3		
	1.1.2.1.3	localDistinguishedName		c:o.3		
	1.1.2.1.3.1	AttributeType		c:m		
	1.1.2.1.3.2	AttributeValue		c:m		
	1.1.2.2	launchPadId		c:o.2		
	1.1.2.2.1	distinguishedName		c:o.4		
	1.1.2.2.1.1	AttributeType		c:m		
	1.1.2.2.1.2	AttributeValue		c:m		
	1.1.2.2.2	nonSpecificForm		c:o.4		
	1.1.2.2.3	localDistinguishedName		c:o.4		
	1.1.2.2.3.1	AttributeType		c:m		
	1.1.2.2.3.2	AttributeValue		c:m		
2	2.1	SpawnerObjectId		m		
	2.1.1	triggerId		m		
	2.1.1.1	distinguishedName		c:o.5		
	2.1.1.1.1	AttributeType		c:m		
	2.1.1.1.2	AttributeValue		c:m		
	2.1.1.2	nonSpecificForm		c:o.5		

Table H.25 *(concluded)*

Index	Subindex	Action field name label	Constraints and values	Status	Support	Additional information
	2.1.1.3	localDistinguishedName		c:o.5		
	2.1.1.3.1	AttributeType		c:m		
	2.1.1.3.2	AttributeValue		c:m		
	2.1.2	CHOICE		m		
	2.1.2.1	threadId		c:o.6		
	2.1.2.1.1	distinguishedName		c:o.7		
	2.1.2.1.1.1	AttributeType		c:m		
	2.1.2.1.1.2	AttributeValue		c:m		
	2.1.2.1.2	nonSpecificForm		c:o.7		
	2.1.2.1.3	localDistinguishedName		c:o.7		
	2.1.2.1.3.1	AttributeType		c:m		
	2.1.2.1.3.2	AttributeValue		c:m		
	2.1.2.2	launchPadId		c:o.6		
	2.1.2.2.1	distinguishedName		c:o.8		
	2.1.2.2.1.1	AttributeType		c:m		
	2.1.2.2.1.2	AttributeValue		c:m		
	2.1.2.2.2	nonSpecificForm		c:o.8		
	2.1.2.2.3	localDistinguishedName		c:o.8		
	2.1.2.2.3.1	AttributeType		c:m		
	2.1.2.2.3.2	AttributeValue		c:m		
3	3.1	TriggerId		m		
	3.1.1	distinguishedName		c:o.9		
	3.1.1.1	AttributeType		c:m		
	3.1.1.2	AttributeValue		c:m		
	3.1.2	nonSpecificForm		c:o.9		
	3.1.3	localDistinguishedName		c:o.9		
	3.1.3.1	AttributeType		c:m		
	3.1.3.2	AttributeValue		c:m		

H.5.6 Notifications

There are no notifications defined for this object class.

H.5.7 Parameters

There are no parameters defined for this object class.

H.6 Statement of conformance to the synchronousLaunchPad object class

Table H.26 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	synchronousLaunchPad	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx5(5)}		

If the answer to the actual class question in the managed object class support Table H.26 is no, the supplier of the implementation shall fill in the actual class support in Table H.27.

Table H.27 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.6.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.28.

Table H.28 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-26/1b) then m else –						

H.6.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.29. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.29 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		–		x		x	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	availabilityStatus	{smi2AttributeID 33}		–		x		x	
4	controlStatus	{smi2AttributeID 34}		–		m		x	
5	launchPadId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx6(6)}		–		m		x	
6	nameBinding	{smi2AttributeID 63}		–		m		x	
7	objectClass	{smi2AttributeID 65}		–		m		x	
8	observedAttributeId	{joint-iso-itu-t ms(9) function(2) part11(11) attribute(7) xx15(15)}		m		m		m	
9	observedObjectInstance	{joint-iso-itu-t ms(9) function(2) part11(11) attribute(7) xx16(16)}		m		m		m	
10	operationalState	{smi2AttributeID 35}		–		x		x	
11	packages	{smi2AttributeID 66}		–		m		x	
12	schedulerName	{smi2AttributeID 67}		–		m		x	
13	usageState	{smi2AttributeID 39}		–		x		x	

Table H.29 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
12	x		x		x		
13	x		x		x		
c1: if not (H-26/1b) then m else –							

H.6.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.6.4 Actions

The supplier of the implementation shall state whether or not the actions specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.30.

Table H.30 – MOCS – Action support

Index	Action type template label	Value of object identifier for action type	Constraints and values	Status	Support	Additional information
1	resume	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx1(1)}		m		
2	suspend	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx2(2)}		m		
3	terminate	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx3(3)}		m		

Table H.31 – MOCS – Action support

Index	Subindex	Action field name label	Constraints and values	Status	Support	Additional information
1	1.1	SpawnerObjectId		m		
	1.1.1	triggerId		m		
	1.1.1.1	distinguishedName		c:o.1		
	1.1.1.1.1	AttributeType		c:m		
	1.1.1.1.2	AttributeValue		c:m		
	1.1.1.2	nonSpecificForm		c:o.1		
	1.1.1.3	localDistinguishedName		c:o.1		
	1.1.1.3.1	AttributeType		c:m		
	1.1.1.3.2	AttributeValue		c:m		
	1.1.2	CHOICE		m		
	1.1.2.1	threadId		c:o.2		
	1.1.2.1.1	distinguishedName		c:o.3		
	1.1.2.1.1.1	AttributeType		c:m		
	1.1.2.1.1.2	AttributeValue		c:m		
	1.1.2.1.2	nonSpecificForm		c:o.3		
	1.1.2.1.3	localDistinguishedName		c:o.3		
	1.1.2.1.3.1	AttributeType		c:m		
	1.1.2.1.3.2	AttributeValue		c:m		
	1.1.2.2	launchPadId		c:o.2		
	1.1.2.2.1	distinguishedName		c:o.4		
	1.1.2.2.1.1	AttributeType		c:m		
	1.1.2.2.1.2	AttributeValue		c:m		
	1.1.2.2.2	nonSpecificForm		c:o.4		
	1.1.2.2.3	localDistinguishedName		c:o.4		
	1.1.2.2.3.1	AttributeType		c:m		
	1.1.2.2.3.2	AttributeValue		c:m		
2	2.1	SpawnerObjectId		m		
	2.1.1	triggerId		m		
	2.1.1.1	distinguishedName		c:o.5		
	2.1.1.1.1	AttributeType		c:m		
	2.1.1.1.2	AttributeValue		c:m		
	2.1.1.2	nonSpecificForm		c:o.5		
	2.1.1.3	localDistinguishedName		c:o.5		
	2.1.1.3.1	AttributeType		c:m		
	2.1.1.3.2	AttributeValue		c:m		
	2.1.2	CHOICE		m		
	2.1.2.1	threadId		c:o.6		
	2.1.2.1.1	distinguishedName		c:o.7		
	2.1.2.1.1.1	AttributeType		c:m		
	2.1.2.1.1.2	AttributeValue		c:m		
	2.1.2.1.2	nonSpecificForm		c:o.7		
	2.1.2.1.3	localDistinguishedName		c:o.7		
	2.1.2.1.3.1	AttributeType		c:m		
	2.1.2.1.3.2	AttributeValue		c:m		
	2.1.2.2	launchPadId		c:o.6		
	2.1.2.2.1	distinguishedName		c:o.8		
	2.1.2.2.1.1	AttributeType		c:m		
	2.1.2.2.1.2	AttributeValue		c:m		
	2.1.2.2.2	nonSpecificForm		c:o.8		
	2.1.2.2.3	localDistinguishedName		c:o.8		
	2.1.2.2.3.1	AttributeType		c:m		
	2.1.2.2.3.2	AttributeValue		c:m		
3	3.1	TriggerId		m		
	3.1.1	distinguishedName		c:o.9		
	3.1.1.1	AttributeType		c:m		
	3.1.1.2	AttributeValue		c:m		
	3.1.2	nonSpecificForm		c:o.9		
	3.1.3	localDistinguishedName		c:o.9		
	3.1.3.1	AttributeType		c:m		
	3.1.3.2	AttributeValue		c:m		

H.6.5 Notifications

There are no notifications defined for this object class.

H.6.6 Parameters

There are no parameters defined for this object class.

H.7 Statement of conformance to the launchScript object class

Table H.32 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	launchScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx7(7)}		

If the answer to the actual class question in the managed object class support Table H.32 is no, the supplier of the implementation shall fill in the actual class support in Table H.33.

Table H.33 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.7.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.34.

Table H.34 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-32/1b) then m else – c2: if H-34/1 then m else –						

H.7.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.35. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.35 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
4	nameBinding	{smi2AttributeID 63}		–		m		x	
5	objectClass	{smi2AttributeID 65}		–		m		x	
6	packages	{smi2AttributeID 66}		–		c2		x	
7	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	

Table H.35 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		

c1: if not (H-32/1b) then m else –
c2: if H.34/2 then m else –

H.7.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.7.4 Actions

There are no actions defined for this object class.

H.7.5 Notifications

There are no notifications defined for this object class.

H.7.6 Parameters

There are no parameters defined for this object class.

H.8 Statement of conformance to the scriptReferencer object class

Table H.36 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	scriptReferencer	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx8(8)}		

If the answer to the actual class question in the managed object class support Table H.36 is no, the supplier of the implementation shall fill in the actual class support in Table H.37.

Table H.37 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.8.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.38.

Table H.38 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-36/1b) then m else –						

H.8.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.39. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.39 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	allomorphs	{smi2AttributeID 50}		x		c1		x	
2	nameBinding	{smi2AttributeID 63}		–		m		x	
3	objectClass	{smi2AttributeID 65}		–		m		x	
4	packages	{smi2AttributeID 66}		–		m		x	

Table H.39 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
c1: if not (H-36/1b) then m else –							

H.8.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.8.4 Actions

There are no actions defined for this object class.

H.8.5 Notifications

There are no notifications defined for this object class.

H.8.6 Parameters

There are no parameters defined for this object class.

H.9 Statement of conformance to the thread object class

Table H.40 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	thread	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx9(9)}		

If the answer to the actual class question in the managed object class support Table H.40 is no, the supplier of the implementation shall fill in the actual class support in Table H.41.

Table H.41 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.9.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.42.

Table H.42 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-40/1b) then m else –						

H.9.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.43. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.43 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	allomorphs	{smi2AttributeID 50}		x		c1		x	
2	executingParameters	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx9(9)}		–		m		x	
3	nameBinding	{smi2AttributeID 63}		–		m		x	
4	objectClass	{smi2AttributeID 65}		–		m		x	
5	operationalState	{smi2AttributeID 35}		–		m		x	
6	packages	{smi2AttributeID 66}		–		m		x	
7	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
8	threadId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx10(10)}		–		m		x	

Table H.43 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
c1: if not (H-40/1b) then m else –							

H.9.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.9.4 Actions

There are no actions defined for this object class.

H.9.5 Notifications

The supplier of the implementation shall state whether or not the notifications specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.44. The supplier of the implementation shall indicate support in terms of the confirmed and non-confirmed modes.

Table H.44 – MOCS – Notification support

					Support		
Index	Notification type template label	Value of object identifier for notification type	Constraints and values	Status	Confirmed	Non-confirmed	Additional information
1	processingErrorAlarm	{smi2Notification 10}		m			

Table H.44 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
1	1.1	additionalInformation	{smi2AttributeID 6}		o		
	1.1.1	identifier	–		c:m		
	1.1.2	significance	–		c:m		
	1.1.3	information	–		c:m		
	1.2	additionalText	{smi2AttributeID 7}		o		
	1.3	backedUpStatus	{smi2AttributeID 11}		o		
	1.4	backUpObject	{smi2AttributeID 40}		o		
	1.4.1	objectName	–		c:o.1		
	1.4.1.1	distinguishedName	–		c:o.2		
	1.4.1.1.1	AttributeType	–		c:m		
	1.4.1.1.2	AttributeValue	–		c:m		
	1.4.1.2	nonSpecificForm	–		c:o.2		
	1.4.1.3	localDistinguishedName	–		c:o.2		
	1.4.1.3.1	AttributeType	–		c:m		
	1.4.1.3.2	AttributeValue	–		c:m		
	1.4.2	noObject	–		c:o.1		
	1.5	correlatedNotifications	{smi2AttributeID 12}		o		
	1.5.1	correlatedNotifications	–		c:m		
	1.5.2	sourceObjectInst	–		c:o		
	1.5.2.1	distinguishedName	–		c:o.3		
	1.5.2.1.1	AttributeType	–		c:m		
	1.5.2.1.2	AttributeValue	–		c:m		
	1.5.2.2	nonSpecificForm	–		c:o.3		
	1.5.2.3	localDistinguishedName	–		c:o.3		
	1.5.2.3.1	AttributeType	–		c:m		
	1.5.2.3.2	AttributeValue	–		c:m		
	1.6	monitoredAttributes	{smi2AttributeID 15}		o		
	1.6.1	attributeId	–		c:m		
	1.6.1.1	globalForm	–		c:o.4		
	1.6.1.2	localForm	–		c:o.4		
	1.6.2	attributeValue	–		c:m		
	1.7	notificationIdentifier	{smi2AttributeID 16}		o		

Table H.44 *(concluded)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
	1.8	perceivedSeverity	{smi2AttributeID 17}		m		
	1.9	probableCause	{smi2AttributeID 18}		m		
	1.9.1	globalValue	–		c:o.5		
	1.9.2	localValue	–		c:o.5		
	1.10	proposedRepairActions	{smi2AttributeID 19}		o		
	1.10.1	OBJECT IDENTIFIER	–		c:o.6		
	1.10.2	INTEGER	–		c:o.6		
	1.11	specificProblems	{smi2AttributeID 27}		o		
	1.11.1	OBJECT IDENTIFIER	–		c:o.7		
	1.11.2	INTEGER	–		c:o.7		
	1.12	stateChangeDefinition	{smi2AttributeID 28}		o		
	1.12.1	attributeID	–		c:m		
	1.12.1.1	globalForm	–		c:o.8		
	1.12.1.2	localForm	–		c:o.8		
	1.12.2	oldAttributeValue	–		c:o		
	1.12.3	newAttributeValue	–		c:m		
	1.13	thresholdInfo	{smi2AttributeID 29}		o		
	1.13.1	triggeredThreshold	–		c:m		
	1.13.1.1	globalForm	–		c:o.9		
	1.13.1.2	localForm	–		c:o.9		
	1.13.2	observedValue	–		c:m		
	1.13.2.1	integer	–		c:o.10		
	1.13.2.2	real	–		c:o.10		
	1.13.3	thresholdLevel	–		c:o		
	1.13.3.1	up	–		c:o.11		
	1.13.3.1.1	high	–		c:m		
	1.13.3.1.1.1	integer	–		c:o.12		
	1.13.3.1.1.2	real	–		c:o.12		
	1.13.3.1.2	low	–		c:o		
	1.13.3.1.2.1	integer	–		c:o.13		
	1.13.3.1.2.2	real	–		c:o.13		
	1.13.3.2	down	–		c:o.11		
	1.13.3.2.1	high	–		c:m		
	1.13.3.2.1.1	integer	–		c:o.14		
	1.13.3.2.1.2	real	–		c:o.14		
	1.13.3.2.2	low	–		c:m		
	1.13.3.2.2.1	integer	–		c:o.15		
	1.13.3.2.2.2	real	–		c:o.15		
	1.13.4	armTime	–		c:o		
	1.14	trendIndication	{smi2AttributeID 30}		o		

H.9.6 Parameters

There are no parameters defined for this object class.

H.10 Statement of conformance to the suspendableThread object class

Table H.45 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	suspendableThread	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx10(10)}		

If the answer to the actual class question in the managed object class support Table H.45 is no, the supplier of the implementation shall fill in the actual class support in Table H.46.

Table H.46 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.10.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.47.

Table H.47 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-45/1b) then m else –						

H.10.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.48. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.48 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	allomorphs	{smi2AttributeID 50}		x		c1		x	
2	controlStatus	{smi2AttributeID 34}		–		m		x	
3	executingParameters	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx9(9)}		–		m		x	
4	nameBinding	{smi2AttributeID 63}		–		m		x	
5	objectClass	{smi2AttributeID 65}		–		m		x	
6	operationalState	{smi2AttributeID 35}		–		m		x	
7	packages	{smi2AttributeID 66}		–		m		x	
8	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
9	threadId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx10(10)}		–		m		x	

Table H.48 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
c1: if not (H-45/1b) then m else –							

H.10.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.10.4 Actions

The supplier of the implementation shall state whether or not the actions specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.49.

Table H.49 – MOCS – Action support

Index	Action type template label	Value of object identifier for action type	Constraints and values	Status	Support	Additional information
1	resume	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx1(1)}		m		
2	suspend	{joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx2(2)}		m		

H.10.5 Notifications

The supplier of the implementation shall state whether or not the notifications specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.51. The supplier of the implementation shall indicate support in terms of the confirmed and non-confirmed modes.

Table H.50 – MOCS – Action support

Index	Subindex	Action field name label	Constraints and values	Status	Support	Additional information
1	1.1	SpawnerObjectId		m		
	1.1.1	triggerId		m		
	1.1.1.1	distinguishedName		c:o.1		
	1.1.1.1.1	AttributeType		c:m		
	1.1.1.1.2	AttributeValue		c:m		
	1.1.1.2	nonSpecificForm		c:o.1		
	1.1.1.3	localDistinguishedName		c:o.1		
	1.1.1.3.1	AttributeType		c:m		
	1.1.1.3.2	AttributeValue		c:m		
	1.1.2	CHOICE		m		
	1.1.2.1	threadId		c:o.2		
	1.1.2.1.1	distinguishedName		c:o.3		
	1.1.2.1.1.1	AttributeType		c:m		
	1.1.2.1.1.2	AttributeValue		c:m		
	1.1.2.1.2	nonSpecificForm		c:o.3		
	1.1.2.1.3	localDistinguishedName		c:o.3		
	1.1.2.1.3.1	AttributeType		c:m		
	1.1.2.1.3.2	AttributeValue		c:m		
	1.1.2.2	launchPadId		c:o.2		
	1.1.2.2.1	distinguishedName		c:o.4		
	1.1.2.2.1.1	AttributeType		c:m		
	1.1.2.2.1.2	AttributeValue		c:m		
	1.1.2.2.2	nonSpecificForm		c:o.4		
	1.1.2.2.3	localDistinguishedName		c:o.4		
	1.1.2.2.3.1	AttributeType		c:m		
	1.1.2.2.3.2	AttributeValue		c:m		
2	2.1	SpawnerObjectId		m		
	2.1.1	triggerId		m		
	2.1.1.1	distinguishedName		c:o.5		
	2.1.1.1.1	AttributeType		c:m		
	2.1.1.1.2	AttributeValue		c:m		
	2.1.1.2	nonSpecificForm		c:o.5		
	2.1.1.3	localDistinguishedName		c:o.5		
	2.1.1.3.1	AttributeType		c:m		
	2.1.1.3.2	AttributeValue		c:m		
	2.1.2	CHOICE		m		
	2.1.2.1	threadId		c:o.6		
	2.1.2.1.1	distinguishedName		c:o.7		
	2.1.2.1.1.1	AttributeType		c:m		
	2.1.2.1.1.2	AttributeValue		c:m		
	2.1.2.1.2	nonSpecificForm		c:o.7		
	2.1.2.1.3	localDistinguishedName		c:o.7		
	2.1.2.1.3.1	AttributeType		c:m		
	2.1.2.1.3.2	AttributeValue		c:m		
	2.1.2.2	launchPadId		c:o.6		
	2.1.2.2.1	distinguishedName		c:o.8		
	2.1.2.2.1.1	AttributeType		c:m		
	2.1.2.2.1.2	AttributeValue		c:m		
	2.1.2.2.2	nonSpecificForm		c:o.8		
	2.1.2.2.3	localDistinguishedName		c:o.8		
	2.1.2.2.3.1	AttributeType		c:m		
	2.1.2.2.3.2	AttributeValue		c:m		

Table H.51 – MOCS – Notification support

Index	Notification type template label	Value of object identifier for notification type	Constraints and values	Status	Support		Additional information
					Confirmed	Non-confirmed	
1	processingErrorAlarm	{smi2Notification 10}		m			

Table H.51 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
1	1.1	additionalInformation	{smi2AttributeID 6}		o		
	1.1.1	identifier	–		c:m		
	1.1.2	significance	–		c:m		
	1.1.3	information	–		c:m		
	1.2	additionalText	{smi2AttributeID 7}		o		
	1.3	backedUpStatus	{smi2AttributeID 11}		o		
	1.4	backUpObject	{smi2AttributeID 40}		o		
	1.4.1	objectName	–		c:o.1		
	1.4.1.1	distinguishedName	–		c:o.2		
	1.4.1.1.1	AttributeType	–		c:m		
	1.4.1.1.2	AttributeValue	–		c:m		
	1.4.1.2	nonSpecificForm	–		c:o.2		
	1.4.1.3	localDistinguishedName	–		c:o.2		
	1.4.1.3.1	AttributeType	–		c:m		
	1.4.1.3.2	AttributeValue	–		c:m		
	1.4.2	noObject	–		c:o.1		
	1.5	correlatedNotifications	{smi2AttributeID 12}		o		
	1.5.1	correlatedNotifications	–		c:m		
	1.5.2	sourceObjectInst	–		c:o		
	1.5.2.1	distinguishedName	–		c:o.3		
	1.5.2.1.1	AttributeType	–		c:m		
	1.5.2.1.2	AttributeValue	–		c:m		
	1.5.2.2	nonSpecificForm	–		c:o.3		
	1.5.2.3	localDistinguishedName	–		c:o.3		
	1.5.2.3.1	AttributeType	–		c:m		
	1.5.2.3.2	AttributeValue	–		c:m		
	1.6	monitoredAttributes	{smi2AttributeID 15}		o		
	1.6.1	attributeId	–		c:m		
	1.6.1.1	globalForm	–		c:o.4		
	1.6.1.2	localForm	–		c:o.4		
	1.6.2	attributeValue	–		c:m		
	1.7	notificationIdentifier	{smi2AttributeID 16}		o		
	1.8	perceivedSeverity	{smi2AttributeID 17}		m		
	1.9	probableCause	{smi2AttributeID 18}		m		
	1.9.1	globalValue	–		c:o.5		
	1.9.2	localValue	–		c:o.5		
	1.10	proposedRepairActions	{smi2AttributeID 19}		o		
	1.10.1	OBJECT IDENTIFIER	–		c:o.6		
	1.10.2	INTEGER	–		c:o.6		
	1.11	specificProblems	{smi2AttributeID 27}		o		
	1.11.1	OBJECT IDENTIFIER	–		c:o.7		
	1.11.2	INTEGER	–		c:o.7		

Table H.51 *(concluded)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
	1.12	stateChangeDefinition	{smi2AttributeID 28}		o		
	1.12.1	attributeID	–		c:m		
	1.12.1.1	globalForm	–		c:o.8		
	1.12.1.2	localForm	–		c:o.8		
	1.12.2	oldAttributeValue	–		c:o		
	1.12.3	newAttributeValue	–		c:m		
	1.13	thresholdInfo	{smi2AttributeID 29}		o		
	1.13.1	triggeredThreshold	–		c:m		
	1.13.1.1	globalForm	–		c:o.9		
	1.13.1.2	localForm	–		c:o.9		
	1.13.2	observedValue	–		c:m		
	1.13.2.1	integer	–		c:o.10		
	1.13.2.2	real	–		c:o.10		
	1.13.3	thresholdLevel	–		c:o		
	1.13.3.1	up	–		c:o.11		
	1.13.3.1.1	high	–		c:m		
	1.13.3.1.1.1	integer	–		c:o.12		
	1.13.3.1.1.2	real	–		c:o.12		
	1.13.3.1.2	low	–		c:o		
	1.13.3.1.2.1	integer	–		c:o.13		
	1.13.3.1.2.2	real	–		c:o.13		
	1.13.3.2	down	–		c:o.11		
	1.13.3.2.1	high	–		c:m		
	1.13.3.2.1.1	integer	–		c:o.14		
	1.13.3.2.1.2	real	–		c:o.14		
	1.13.3.2.2	low	–		c:m		
	1.13.3.2.2.1	integer	–		c:o.15		
	1.13.3.2.2.2	real	–		c:o.15		
	1.13.4	armTime	–		c:o		
	1.14	trendIndication	{smi2AttributeID 30}		o		

H.10.6 Parameters

There are no parameters defined for this object class.

H.11 Statement of conformance to the eventDiscriminationCounter object class

Table H.52 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	eventDiscriminationCounter	{joint-iso-itu-t ms(9) ms(9) function(2) part21(21) managedObjectClass(3) xx11(11)}		

If the answer to the actual class question in the managed object class support Table H.52 is no, the supplier of the implementation shall fill in the actual class support in Table H.53.

Table H.53 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.11.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.54.

Table H.54 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	availabilityStatusPackage	{smi2Package 22}		c2		
3	counterAlarmPackage	{joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx15(15)}		c3		
4	dailyScheduling	{smi2Package 25}		o		
5	duration	{smi2Package 26}		c4		
6	externalScheduler	{smi2Package 27}		o		
7	packagesPackage	{smi2Package 16}		c5		
8	weeklyScheduling	{smi2Package 29}		o		

c1: if not (H-52/1b) then m else –

c2: if "any of the scheduling packages, (duration, weekly scheduling, external) are present" then m else –

c3: if "a counter is of finite size and a notification is triggered by a capacity alarm threshold" then m else –

c4: if "the discriminator function is scheduled to start at a specified time and stop at either a specified time or function continuously" then m else –

c5: if H-54/1 or H-54/2 or H-54/3 or H-54/4 or H-54/5 or H-54/6 or H-54/8 then m else –

H.11.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.55. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

H.11.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.11.4 Actions

There are no actions defined for this object class.

H.11.5 Notifications

The supplier of the implementation shall state whether or not the notifications specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.56. The supplier of the implementation shall indicate support in terms of the confirmed and non-confirmed modes.

Table H.55 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	availabilityStatus	{smi2AttributeID 33}		–		c2		x	
4	capacityAlarmThreshold	{smi2AttributeID 52}		c3		c3		c3	
5	counter	{smi2AttributeID 88}		–		m		x	
6	discriminatorConstruct	{smi2AttributeID 56}		m		m		m	
7	discriminatorId	{smi2AttributeID 1}		–		m		x	
8	intervalsOfDay	{smi2AttributeID 57}		o		o		o	
9	maxCounterSize	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx12(12)}		–		m		x	
10	nameBinding	{smi2AttributeID 63}		–		m		x	
11	objectClass	{smi2AttributeID 65}		–		m		x	
12	operationalState	{smi2AttributeID 35}		–		m		x	
13	packages	{smi2AttributeID 66}		–		c4		x	
14	schedulerName	{smi2AttributeID 67}		–		o		x	
15	startTime	{smi2AttributeID 68}		c5		c5		c5	
16	stopTime	{smi2AttributeID 69}		c5		c5		c5	
17	weekMask	{smi2AttributeID 71}		o		o		o	

Table H.55 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	c3		c3		x		
5	x		x		x		
6	x		x		m		
7	x		x		x		
8	o		o		o		
9	x		x		x		
10	x		x		x		
11	x		x		x		
12	x		x		x		
13	x		x		x		
14	x		x		x		
15	x		x		x		
16	x		x		c5		
17	o		o		o		

c1: if not (H-52/1b) then m else –
c2: if H-54/2 then m else –
c3: if H-54/3 then m else –
c4: if H-54/7 then m else –
c5: if H-54/5 then m else –

Table H.56 – MOCS – Notification support

Index	Notification type template label	Value of object identifier for notification type	Constraints and values	Status	Support		Additional information
					Confirmed	Non-confirmed	
1	attributeValueChange	{smi2Notification 1}		m			
2	objectCreation	{smi2Notification 6}		m			
3	objectDeletion	{smi2Notification 7}		m			
4	processingErrorAlarm	{smi2Notification 10}		m			
5	stateChange	{smi2Notification 14}		m			

Table H.56 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
1	1.1	additionalInformation	{smi2AttributeID 6}		o		
	1.1.1	identifier	–		c:m		
	1.1.2	significance	–		c:m		
	1.1.3	information	–		c:m		
	1.2	additionalText	{smi2AttributeID 7}		o		
	1.3	attributeIdentifierList	{smi2AttributeID 8}		o		
	1.3.1	globalForm	–		c:o.1		
	1.3.2	localForm	–		c:o.1		
	1.4	attributeValueChangeDefinition	{smi2AttributeID 10}		m		
	1.4.1	attributeID	–		m		
	1.4.1.1	globalForm	–		c:o.2		
	1.4.1.2	localForm	–		c:o.2		
	1.4.2	oldAttributeValue	–		o		
	1.4.3	newAttributeValue	–		m		
	1.5	correlatedNotifications	{smi2AttributeID 12}		o		
	1.5.1	correlatedNotifications	–		c:m		
	1.5.2	sourceObjectInst	–		c:o		
	1.5.2.1	distinguishedName	–		c:o.3		
	1.5.2.1.1	AttributeType	–		c:m		
	1.5.2.1.2	AttributeValue	–		c:m		
	1.5.2.2	nonSpecificForm	–		c:o.3		
	1.5.2.3	localDistinguishedName	–		c:o.3		
	1.5.2.3.1	AttributeType	–		c:m		
	1.5.2.3.2	AttributeValue	–		c:m		
	1.6	notificationIdentifier	{smi2AttributeID 16}		o		
	1.7	sourceIndicator	{smi2AttributeID 26}		o		

Table H.56 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
2	2.1	additionalInformation	{smi2AttributeID 6}		o		
	2.1.1	identifier	–		c:m		
	2.1.2	significance	–		c:m		
	2.1.3	information	–		c:m		
	2.2	additionalText	{smi2AttributeID 7}		o		
	2.3	attributeList	{smi2AttributeID 9}		o		
	2.3.1	attributeId	–		c:m		
	2.3.1.1	globalForm	–		c:o.4		
	2.3.1.2	localForm	–		c:o.4		
	2.3.2	attributeValue	–		c:m		
	2.4	correlatedNotifications	{smi2AttributeID 12}		o		
	2.4.1	correlatedNotifications	–		c:m		
	2.4.2	sourceObjectInst	–		c:o		
	2.4.2.1	distinguishedName	–		c:o.5		
	2.4.2.1.1	AttributeType	–		c:m		
	2.4.2.1.2	AttributeValue	–		c:m		
	2.4.2.2	nonSpecificForm	–		c:o.5		
	2.4.2.3	localDistinguishedName	–		c:o.5		
	2.4.2.3.1	AttributeType	–		c:m		
	2.4.2.3.2	AttributeValue	–		c:m		
	2.5	notificationIdentifier	{smi2AttributeID 16}		o		
	2.6	sourceIndicator	{smi2AttributeID 26}		o		
3	3.1	additionalInformation	{smi2AttributeID 6}		o		
	3.1.1	identifier	–		c:m		
	3.1.2	significance	–		c:m		
	3.1.3	information	–		c:m		
	3.2	additionalText	{smi2AttributeID 7}		o		
	3.3	attributeList	{smi2AttributeID 9}		o		
	3.3.1	attributeId	–		c:m		
	3.3.1.1	globalForm	–		c:o.6		
	3.3.1.2	localForm	–		c:o.6		
	3.3.2	attributeValue	–		c:m		
	3.4	correlatedNotifications	{smi2AttributeID 12}		o		
	3.4.1	correlatedNotifications	–		c:m		
	3.4.2	sourceObjectInst	–		c:o		
	3.4.2.1	distinguishedName	–		c:o.7		
	3.4.2.1.1	AttributeType	–		c:m		
	3.4.2.1.2	AttributeValue	–		c:m		
	3.4.2.2	nonSpecificForm	–		c:o.7		
	3.4.2.3	localDistinguishedName	–		c:o.7		
	3.4.2.3.1	AttributeType	–		c:m		
	3.4.2.3.2	AttributeValue	–		c:m		
	3.5	notificationIdentifier	{smi2AttributeID 16}		o		
	3.6	sourceIndicator	{smi2AttributeID 26}		o		
4	4.1	additionalInformation	{smi2AttributeID 6}		o		
	4.1.1	identifier	–		c:m		
	4.1.2	significance	–		c:m		
	4.1.3	information	–		c:m		
	4.2	additionalText	{smi2AttributeID 7}		o		
	4.3	backedUpStatus	{smi2AttributeID 11}		o		
	4.4	backUpObject	{smi2AttributeID 40}		o		
	4.4.1	objectName	–		c:o.8		
	4.4.1.1	distinguishedName	–		c:o.9		
	4.4.1.1.1	AttributeType	–		c:m		

Table H.56 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
	4.4.1.1.2	AttributeValue	–		c:m		
	4.4.1.2	nonSpecificForm	–		c:o.9		
	4.4.1.3	localDistinguishedName	–		c:o.9		
	4.4.1.3.1	AttributeType	–		c:m		
	4.4.1.3.2	AttributeValue	–		c:m		
	4.4.2	noObject	–		c:o.8		
	4.5	correlatedNotifications	{smi2AttributeID 12}		o		
	4.5.1	correlatedNotifications	–		c:m		
	4.5.2	sourceObjectInst	–		c:o		
	4.5.2.1	distinguishedName	–		c:o.10		
	4.5.2.1.1	AttributeType	–		c:m		
	4.5.2.1.2	AttributeValue	–		c:m		
	4.5.2.2	nonSpecificForm	–		c:o.10		
	4.5.2.3	localDistinguishedName	–		c:o.10		
	4.5.2.3.1	AttributeType	–		c:m		
	4.5.2.3.2	AttributeValue	–		c:m		
	4.6	monitoredAttributes	{smi2AttributeID 15}		o		
	4.6.1	attributeId	–		c:m		
	4.6.1.1	globalForm	–		c:o.11		
	4.6.1.2	localForm	–		c:o.11		
	4.6.2	attributeValue	–		c:m		
	4.7	notificationIdentifier	{smi2AttributeID 16}		o		
	4.8	perceivedSeverity	{smi2AttributeID 17}		m		
	4.9	probableCause	{smi2AttributeID 18}		m		
	4.9.1	globalValue	–		c:o.12		
	4.9.2	localValue	–		c:o.12		
	4.10	proposedRepairActions	{smi2AttributeID 19}		o		
	4.10.1	OBJECT IDENTIFIER	–		c:o.13		
	4.10.2	INTEGER	–		c:o.13		
	4.11	specificProblems	{smi2AttributeID 27}		o		
	4.11.1	OBJECT IDENTIFIER	–		c:o.14		
	4.11.2	INTEGER	–		c:o.14		
	4.12	stateChangeDefinition	{smi2AttributeID 28}		o		
	4.12.1	attributeID	–		c:m		
	4.12.1.1	globalForm	–		c:o.15		
	4.12.1.2	localForm	–		c:o.15		
	4.12.2	oldAttributeValue	–		c:o		
	4.12.3	newAttributeValue	–		c:m		
	4.13	thresholdInfo	{smi2AttributeID 29}		o		
	4.13.1	triggeredThreshold	–		c:m		
	4.13.1.1	globalForm	–		c:o.16		
	4.13.1.2	localForm	–		c:o.16		
	4.13.2	observedValue	–		c:m		
	4.13.2.1	integer	–		c:o.17		
	4.13.2.2	real	–		c:o.17		
	4.13.3	thresholdLevel	–		c:o		
	4.13.3.1	up	–		c:o.18		
	4.13.3.1.1	high	–		c:m		
	4.13.3.1.1.1	integer	–		c:o.19		
	4.13.3.1.1.2	real	–		c:o.19		
	4.13.3.1.2	low	–		c:o		
	4.13.3.1.2.1	integer	–		c:o.20		
	4.13.3.1.2.2	real	–		c:o.20		

Table H.56 *(concluded)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
	4.13.3.2	down	–		c:o.18		
	4.13.3.2.1	high	–		c:m		
	4.13.3.2.1.1	integer	–		c:o.21		
	4.13.3.2.1.2	real	–		c:o.21		
	4.13.3.2.2	low	–		c:m		
	4.13.3.2.2.1	integer	–		c:o.22		
	4.13.3.2.2.2	real	–		c:o.22		
	4.13.4	armTime	–		c:o		
	4.14	trendIndication	{smi2AttributeID 30}		o		
5	5.1	additionalInformation	{smi2AttributeID 6}		o		
	5.1.1	identifier	–		c:m		
	5.1.2	significance	–		c:m		
	5.1.3	information	–		c:m		
	5.2	additionalText	{smi2AttributeID 7}		o		
	5.3	attributeIdentifierList	{smi2AttributeID 8}		o		
	5.3.1	globalForm	–		c:o.23		
	5.3.2	localForm	–		c:o.23		
	5.4	correlatedNotifications	{smi2AttributeID 12}		o		
	5.4.1	correlatedNotifications	–		c:m		
	5.4.2	sourceObjectInst	–		c:o		
	5.4.2.1	distinguishedName	–		c:o.24		
	5.4.2.1.1	AttributeType	–		c:m		
	5.4.2.1.2	AttributeValue	–		c:m		
	5.4.2.2	nonSpecificForm	–		c:o.24		
	5.4.2.3	localDistinguishedName	–		c:o.24		
	5.4.2.3.1	AttributeType	–		c:m		
	5.4.2.3.2	AttributeValue	–		c:m		
	5.5	notificationIdentifier	{smi2AttributeID 16}		o		
	5.6	sourceIndicator	{smi2AttributeID 26}		o		
	5.7	stateChangeDefinition	{smi2AttributeID 28}		m		
	5.7.1	attributeID	–		m		
	5.7.1.1	globalForm	–		c:o.25		
	5.7.1.2	localForm	–		c:o.25		
	5.7.2	oldAttributeValue	–		o		
	5.7.3	newAttributeValue	–		m		

H.11.6 Parameters

There are no parameters defined for this object class.

H.12 Statement of conformance to the cmipCS object class

Table H.57 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	cmipCS	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx18(18)}		

If the answer to the actual class question in the managed object class support Table H.57 is no, the supplier of the implementation shall fill in the actual class support in Table H.58.

Table H.58 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.12.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.59.

Table H.59 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		m		
c1: if not (H-57/1b) then m else –						

H.12.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.60. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.60 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	aetitle	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx20(20)}		–		m		x	
3	allomorphs	{smi2AttributeID 50}		x		c1		x	
4	commandSequencerId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx2(2)}		–		m		x	
5	nameBinding	{smi2AttributeID 63}		–		m		x	
6	objectClass	{smi2AttributeID 65}		–		m		x	
7	operationalState	{smi2AttributeID 35}		–		m		x	
8	packages	{smi2AttributeID 66}		–		m		x	

Table H.60 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
c1: if not (H-57/1b) then m else –							

H.12.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.12.4 Actions

There are no actions defined for this object class.

H.12.5 Notifications

The supplier of the implementation shall state whether or not the notifications specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.61. The supplier of the implementation shall indicate support in terms of the confirmed and non-confirmed modes.

Table H.61 – MOCS – Notification support

					Support		
Index	Notification type template label	Value of object identifier for notification type	Constraints and values	Status	Confirmed	Non-confirmed	Additional information
1	objectCreation	{smi2Notification 6}		m			
2	objectDeletion	{smi2Notification 7}		m			
3	stateChange	{smi2Notification 14}		m			

Table H.61 *(continued)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
1	1.1	additionalInformation	{smi2AttributeID 6}		o		
	1.1.1	identifier	–		c:m		
	1.1.2	significance	–		c:m		
	1.1.3	information	–		c:m		
	1.2	additionalText	{smi2AttributeID 7}		o		
	1.3	attributeList	{smi2AttributeID 9}		o		
	1.3.1	attributeId	–		c:m		
	1.3.1.1	globalForm	–		c:o.1		
	1.3.1.2	localForm	–		c:o.1		
	1.3.2	attributeValue	–		c:m		
	1.4	correlatedNotifications	{smi2AttributeID 12}		o		
	1.4.1	correlatedNotifications	–		c:m		
	1.4.2	sourceObjectInst	–		c:o		
	1.4.2.1	distinguishedName	–		c:o.2		
	1.4.2.1.1	AttributeType	–		c:m		
	1.4.2.1.2	AttributeValue	–		c:m		
	1.4.2.2	nonSpecificForm	–		c:o.2		

Table H.61 *(concluded)*

Index	Subindex	Notification field name label	Value of object identifier of attribute type associated with field	Constraints and values	Status	Support	Additional information
	1.4.2.3	localDistinguishedName	–		c:o.2		
	1.4.2.3.1	AttributeType	–		c:m		
	1.4.2.3.2	AttributeValue	–		c:m		
	1.5	notificationIdentifier	{smi2AttributeID 16}		o		
	1.6	sourceIndicator	{smi2AttributeID 26}		o		
2	2.1	additionalInformation	{smi2AttributeID 6}		o		
	2.1.1	identifier	–		c:m		
	2.1.2	significance	–		c:m		
	2.1.3	information	–		c:m		
	2.2	additionalText	{smi2AttributeID 7}		o		
	2.3	attributeList	{smi2AttributeID 9}		o		
	2.3.1	attributeId	–		c:m		
	2.3.1.1	globalForm	–		c:o.3		
	2.3.1.2	localForm	–		c:o.3		
	2.3.2	attributeValue	–		c:m		
	2.4	correlatedNotifications	{smi2AttributeID 12}		o		
	2.4.1	correlatedNotifications	–		c:m		
	2.4.2	sourceObjectInst	–		c:o		
	2.4.2.1	distinguishedName	–		c:o.4		
	2.4.2.1.1	AttributeType	–		c:m		
	2.4.2.1.2	AttributeValue	–		c:m		
	2.4.2.2	nonSpecificForm	–		c:o.4		
	2.4.2.3	localDistinguishedName	–		c:o.4		
	2.4.2.3.1	AttributeType	–		c:m		
	2.4.2.3.2	AttributeValue	–		c:m		
	2.5	notificationIdentifier	{smi2AttributeID 16}		o		
	2.6	sourceIndicator	{smi2AttributeID 26}		o		
3	3.1	additionalInformation	{smi2AttributeID 6}		o		
	3.1.1	identifier	–		c:m		
	3.1.2	significance	–		c:m		
	3.1.3	information	–		c:m		
	3.2	additionalText	{smi2AttributeID 7}		o		
	3.3	attributeIdentifierList	{smi2AttributeID 8}		o		
	3.3.1	globalForm	–		c:o.5		
	3.3.2	localForm	–		c:o.5		
	3.4	correlatedNotifications	{smi2AttributeID 12}		o		
	3.4.1	correlatedNotifications	–		c:m		
	3.4.2	sourceObjectInst	–		c:o		
	3.4.2.1	distinguishedName	–		c:o.6		
	3.4.2.1.1	AttributeType	–		c:m		
	3.4.2.1.2	AttributeValue	–		c:m		
	3.4.2.2	nonSpecificForm	–		c:o.6		
	3.4.2.3	localDistinguishedName	–		c:o.6		
	3.4.2.3.1	AttributeType	–		c:m		
	3.4.2.3.2	AttributeValue	–		c:m		
	3.5	notificationIdentifier	{smi2AttributeID 16}		o		
	3.6	sourceIndicator	{smi2AttributeID 26}		o		
	3.7	stateChangeDefinition	{smi2AttributeID 28}		m		
	3.7.1	attributeID	–		m		
	3.7.1.1	globalForm	–		c:o.7		
	3.7.1.2	localForm	–		c:o.7		
	3.7.2	oldAttributeValue	–		o		
	3.7.3	newAttributeValue	–		m		

H.12.6 Parameters

There are no parameters defined for this object class.

H.13 Statement of conformance to the cmisScript object class

Table H.62 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	cmisScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx12(12)}		

If the answer to the actual class question in the managed object class support Table H.62 is no, the supplier of the implementation shall fill in the actual class support in Table H.63.

Table H.63 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.13.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.64.

Table H.64 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-62/1b) then m else – c2: if H-64/1 then m else –						

H.13.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.65. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.65 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
4	nameBinding	{smi2AttributeID 63}		–		m		x	
5	objectClass	{smi2AttributeID 65}		–		m		x	
6	packages	{smi2AttributeID 66}		–		c2		x	
7	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	

Table H.65 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
c1: if not(H-62/1b) then m else – c2: if H-64/2 then m else –							

H.13.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.13.4 Actions

There are no actions defined for this object class.

H.13.5 Notifications

There are no notifications defined for this object class.

H.13.6 Parameters

There are no parameters defined for this object class.

H.14 Statement of conformance to the getCmisScript object class

Table H.66 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	getCmisScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx13(13)}		

If the answer to the actual class question in the managed object class support Table H.66 is no, the supplier of the implementation shall fill in the actual class support in Table H.67.

Table H.67 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.14.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.68.

Table H.68 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-66/1b) then m else – c2: if H-68/1 then m else –						

H.14.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.69. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.69 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	attributeIdentifierList	{smi2AttributeID 8}		m		m		m	
4	baseManagedObjectId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx13(13)}		–		m		x	
5	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
6	filter	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx15(15)}		m		m		m	
7	nameBinding	{smi2AttributeID 63}		–		m		x	
8	objectClass	{smi2AttributeID 65}		–		m		x	
9	packages	{smi2AttributeID 66}		–		c2		x	
10	scope	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx14(14)}		m		m		m	
11	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
12	synchronization	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx16(16)}		m		m		m	

Table H.69 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	m		m		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
12	x		x		x		
c1: if not (H-66/1b) then m else – c2: if H-68/2 then m else –							

H.14.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.14.4 Actions

There are no actions defined for this object class.

H.14.5 Notifications

There are no notifications defined for this object class.

H.14.6 Parameters

There are no parameters defined for this object class.

H.15 Statement of conformance to the setCmisScript object class

Table H.70 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	setCmisScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx14(14)}		

If the answer to the actual class question in the managed object class support Table H.70 is no, the supplier of the implementation shall fill in the actual class support in Table H.71.

Table H.71 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.15.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.72.

Table H.72 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-70/1b) then m else – c2: if H-72/1 then m else –						

H.15.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.73. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.73 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	baseManagedObjectId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx13(13)}		–		m		x	
4	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
5	filter	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx15(15)}		m		m		m	
6	modificationList	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx17(17)}		m		m		m	
7	nameBinding	{smi2AttributeID 63}		–		m		x	
8	objectClass	{smi2AttributeID 65}		–		m		x	
9	packages	{smi2AttributeID 66}		–		c2		x	
10	scope	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx14(14)}		m		m		m	
11	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
12	synchronization	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx16(16)}		m		m		m	

Table H.73 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	m		m		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
12	x		x		x		
c1: if not (H-70/1b) then m else – c2: if H-72/2 then m else –							

H.15.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.15.4 Actions

There are no actions defined for this object class.

H.15.5 Notifications

There are no notifications defined for this object class.

H.15.6 Parameters

There are no parameters defined for this object class.

H.16 Statement of conformance to the actionCmisScript object class

Table H.74 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	actionCmisScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx15(15)}		

If the answer to the actual class question in the managed object class support Table H.74 is no, the supplier of the implementation shall fill in the actual class support in Table H.75.

Table H.75 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.16.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.76.

Table H.76 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-74/1b) then m else – c2: if H-76/1 then m else –						

H.16.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.77. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.77 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	baseManagedObjectId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx13(13)}		–		m		x	
4	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
5	filter	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx15(15)}		m		m		m	
6	nameBinding	{smi2AttributeID 63}		–		m		x	
7	objectClass	{smi2AttributeID 65}		–		m		x	
8	packages	{smi2AttributeID 66}		–		c2		x	
9	scope	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx14(14)}		m		m		m	
10	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
11	synchronization	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx16(16)}		m		m		m	

Table H.77 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
c1: if not (H-74/1b) then m else – c2: if H-76/2 then m else –							

H.16.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.16.4 Actions

There are no actions defined for this object class.

H.16.5 Notifications

There are no notifications defined for this object class.

H.16.6 Parameters

There are no parameters defined for this object class.

H.17 Statement of conformance to the createCmisScript object class

Table H.78 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	createCmisScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx16(16)}		

If the answer to the actual class question in the managed object class support Table H.78 is no, the supplier of the implementation shall fill in the actual class support in Table H.79.

Table H.79 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.17.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the “Support” and “Additional information” columns in Table H.80.

Table H.80 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	managedObjectInstancePackage	{joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx16(16)}		c2		
3	packagesPackage	{smi2Package 16}		c3		
4	referenceObjectInstancePackage	{joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx18(18)}		c4		
5	superiourObjectInstancePackage	{joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx17(17)}		c5		

c1: if not (H-78/1b) then m else –

c2: if “the superiourObjectInstancePackage is not present” then m else –

c3: if H-80/1 or H-80/2 or H-80/4 or H-80/5 then m else –

c4: if “the manager has the specified value” then m else –

c5: if “the managedObjectInstance Package is not present” then m else –

H.17.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the “Support” and “Additional information” columns in Table H.81. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.81 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	attributeList	{smi2AttributeID 9}		m		m		m	
4	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
5	managedObjectInstance	{smi2AttributeID 61}		c2		c2		c2	
6	nameBinding	{smi2AttributeID 63}		–		m		x	
7	objectClass	{smi2AttributeID 65}		–		m		x	
8	packages	{smi2AttributeID 66}		–		c3		x	
9	referenceObjectInstance	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx19(19)}		c4		c4		c4	
10	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
11	superiourObjectInstance	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx18(18)}		c5		c5		c5	

Table H.81 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	m		m		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		

c1: if not (H-78/1b) then m else –
c2: if H-80/2 then m else –
c3: if H-80/3 then m else –
c4: if H-80/4 then m else –
c5: if H-80/5 then m else –

H.17.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.17.4 Actions

There are no actions defined for this object class.

H.17.5 Notifications

There are no notifications defined for this object class.

H.17.6 Parameters

There are no parameters defined for this object class.

H.18 Statement of conformance to the deleteCmisScript object class

Table H.82 – MOCS – Managed object class support

Index	Managed object class template label	Value of object identifier for class	Support of all mandatory features	Is the actual class the same as the managed object class to which conformance is claimed? (Y/N)
1	deleteCmisScript	{joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx17(17)}		

If the answer to the actual class question in the managed object class support Table H.82 is no, the supplier of the implementation shall fill in the actual class support in Table H.83.

Table H.83 – MOCS – Actual class support

Index	Actual managed object class template label	Value of object identifier for actual class	Additional information
1			
2			

H.18.1 Packages

The supplier of the implementation shall state whether or not the conditional packages specified by this class are supported by an instance of this class, in the "Support" and "Additional information" columns in Table H.84.

Table H.84 – MOCS – Package support

Index	Package template label	Value of object identifier for package	Constraints and values	Status	Support	Additional information
1	allomorphicPackage	{smi2Package 17}		c1		
2	packagesPackage	{smi2Package 16}		c2		
c1: if not (H-82/1b) then m else – c2: if H-84/1 then m else –						

H.18.2 Attributes

The supplier of the implementation shall state whether or not the attributes specified by all packages instantiated in a managed object of this class are supported, in the "Support" and "Additional information" columns in Table H.85. The supplier of the implementation shall indicate support for each of the operations for each attribute supported.

Table H.85 – MOCS – Attribute support

				Set by create		Get		Replace	
Index	Attribute template label	Value of object identifier for attribute	Constraints and values	Status	Support	Status	Support	Status	Support
1	administrativeState	{smi2AttributeID 31}		m		m		m	
2	allomorphs	{smi2AttributeID 50}		x		c1		x	
3	baseManagedObjectId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx13(13)}		–		m		x	
4	executionResultType	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)}		–		m		x	
5	filter	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx15(15)}		m		m		m	
6	nameBinding	{smi2AttributeID 63}		–		m		x	
7	objectClass	{smi2AttributeID 65}		–		m		x	
8	packages	{smi2AttributeID 66}		–		c2		x	
9	scope	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx14(14)}		m		m		m	
10	scriptId	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)}		–		m		x	
11	synchronization	{joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx16(16)}		m		m		m	

Table H.85 *(concluded)*

	Add		Remove		Set to default		
Index	Status	Support	Status	Support	Status	Support	Additional information
1	x		x		x		
2	x		x		x		
3	x		x		x		
4	x		x		x		
5	x		x		x		
6	x		x		x		
7	x		x		x		
8	x		x		x		
9	x		x		x		
10	x		x		x		
11	x		x		x		
c1: if not (H-82/1b) then m else – c2: if H-84/2 then m else –							

H.18.3 Attribute groups

There are no attribute groups defined for the managed object class.

H.18.4 Actions

There are no actions defined for this object class.

H.18.5 Notifications

There are no notifications defined for this object class.

H.18.6 Parameters

There are no parameters defined for this object class.

SÉRIES 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